A DATA MANAGEMENT SYSTEM FOR A MICROCOMPUTER BASED TRACTOR DATA ACQUISITION SYSTEM

By

DOUGLAS ROBERT DEVOE

Bachelor of Science

in Agricultural Engineering

Oklahoma State University

Stillwater, Oklahoma

1980

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the degree of
MASTER OF SCIENCE
December, 1982

A DATA MANAGEMENT SYSTEM FOR A MICROCOMPUTER BASED TRACTOR DATA ACQUISITION SYSTEM

Thesis Approved:

David G Batchelder
Thesis Adviser

Richard W. Whitney

Norman N Durham
Dean of the Graduate College

ACKNOWLEDGMENTS

The author wishes to express gratitude and appreciation to his major adviser, Professor David G. Batchelder, for his guidance, encouragement, and patience throughout this work. The author also expresses appreciation to all of the faculty and staff of the Agricultural Engineering Department and to the Oklahoma Wheat Commision, whose financial support enabled the purchase of much of the equipment used in the system.

My wife, Janet, also deserves credit for help in formatting the thesis as well as her endurance after hours. Finally, my parents, James and Ruth DeVoe, have played a role in this work since their constant "nudging" helped to generate motivation at perhaps the most crucial time of development.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER I

INTRODUCTION

Measuring the performance of an agricultural tractor can be a complicated task. The flow rate of the fuel, controlled by the throttle and governor, is constantly changing. The force required to pull an implement through the soil also changes constantly--in direction as well as magnitude. Different soil conditions cause different rolling resistances as do sloped soil surfaces. In many instances, the environment of the tractor is extremely harsh, caused by bumps, jolts, dust, and heat. All of these are considerations when attempting to measure tractor performance.

An instrument many researchers are using to help achieve this performance measurement is the microcomputer. John Deere Harvester Works has done extensive research in this area (1). Many universities also use microcomputers to measure tractor performance. For example, Clemson, Auburn, and the University of Tennessee have built microcomputer based tractor performance instruments (2) (3) (4).

The Agricultural Engineering Department at Oklahoma State University also uses a microcomputer to help measure tractor performance (5). The building of the original microcomputer data acquisition system began in the fall of 1979. At that time, the Agricultural Engineering Department

arranged an agreement with the Electrical Engineering Department to develop a data acquisition system based on an Intel 8010 microcomputer. The system was to record data on cassette tape so it could be transferred to the University Computer, an IBM 370. Since the Electrical Engineering Department fell behind schedule, the Agricultural Engineering Department decided to develop their own tractor performance monitor. This system, a simpler version than the one intended by Electrical Engineering, was developed within about 3 months.

The Agricultural Engineering Department's tractor performance monitor uses an Aim 65 microcomputer to measure 5 frequencies: fuel flow, pull, ground speed, wheel speed, and drawbar pull. Pull is measured using strain guages, the output voltage being converted to a frequency which the Aim 65 measures. The engine rpm measurement uses a magnet attached to the engine crankshaft. The magnet passes a Hall Effect switch to generate a 5 volt frequency. The remaining three transducers use rotary shaft encoders.

The Aim 65 measures each of the 5 variables in sequence, prints the results on paper tape, and repeats this 10 times. Then the data, in hexadecimal format, is recorded on cassette tape. These 50 numbers, 10 replications of 5 numbers, are hereinafter referred to as an experimental unit.

By recording the experimental unit on cassette tape, the data can be more quickly analyzed. If the data were only

available on paper, the experimenter would have to type the data into the computer in order to use the computer to help analyze the data. Typing the data is time consuming, and human error is introduced. By recording the data on cassette tape, it can be quickly transferred to another computer for analysis.

As the microcomputer data acquisition system creates large quantities of data, there are advantages to organizing the data so any part of it can be quickly accessed by a computer at a later time. For example, if all the data on moldboard plows used in June of 1981 were required, without a computerized data management system, the researcher must search by hand, or write a customized program to search through the data. As the size of the database increases, the time saved by having the computer system to perform the search will become more important. Thus, for every microcomputer based data acquisition system designed to acquire large amounts of data, there should be a complimentary database management system to accommodate the data. The scope of this thesis is to provide such a database management system for the previously described microcomputer based tractor performance data acquisition system.

CHAPTER II

OBJECTIVE

The objective of this thesis is to develop computer software to perform managerial operations on data recorded on cassette tape by a microcomputer based tractor performance monitor. The managerial operations include the ability to do the following:

1. Read the data from cassette and allow user verification.
2. Catalog the data and record it on cartridge tape.
3. Edit any part of the data recorded on cartridge tape.
4. Transfer all or selected parts of the data from cartridge tape to the University Computer.

CHAPTER III

LITERATURE REVIEW

Introduction

The invention of the transistor around 1950 was the begining of the modern computer oriented society. The transistor reduced the amount of space required per unit of information stored by the computer by many orders of magnitude. As a consequence, one of the principal uses of computers today is for information storage and retrieval. In fact, there is an entire area of computer science devoted to improving techniques of storage and retrieval called Database Management. Many organizations have large main-frame computers which handle millions of bytes of information via software programs called Database Management Systems or DBMS.

An example of an organization currently using a DBMS is the National Institutes of Health (6). Within the NIH, the Division of Computer Research And Technology maintains and operates a central computing facility which serves over 8,000 users. One of the DBMS applications is a material management system for purchasing, inventory, and related functions. Another DBMS keeps information on the several thousand users, students in training courses, and the hundreds

of terminals rented by users.

The Social Security Administration has a DBMS capable of handling an enormous amount of information. One database contains records of individuals on the Retirement and Survivor's Insurance Program (6). This database contains 33 billion bytes, which constitutes 35 million records. Likewise, another database keeps track of individuals on the Supplemental Security Income Program. This database handles 7 billion bytes, or 5 million records (6).

Within the USDA, the Food Safety And Quality Service uses a DBMS to store mailing label information used to mail USDA regulations and circulars. This database contains about 16 megabytes and has been estimated to save several thousand dollars a year on return mail. The Food Safety and Quality Service also maintains a chemical compounds and packaging database that keeps track of approved chemicals near food producers. The 16 megabyte database is updated daily--during a typical month about a thousand times.

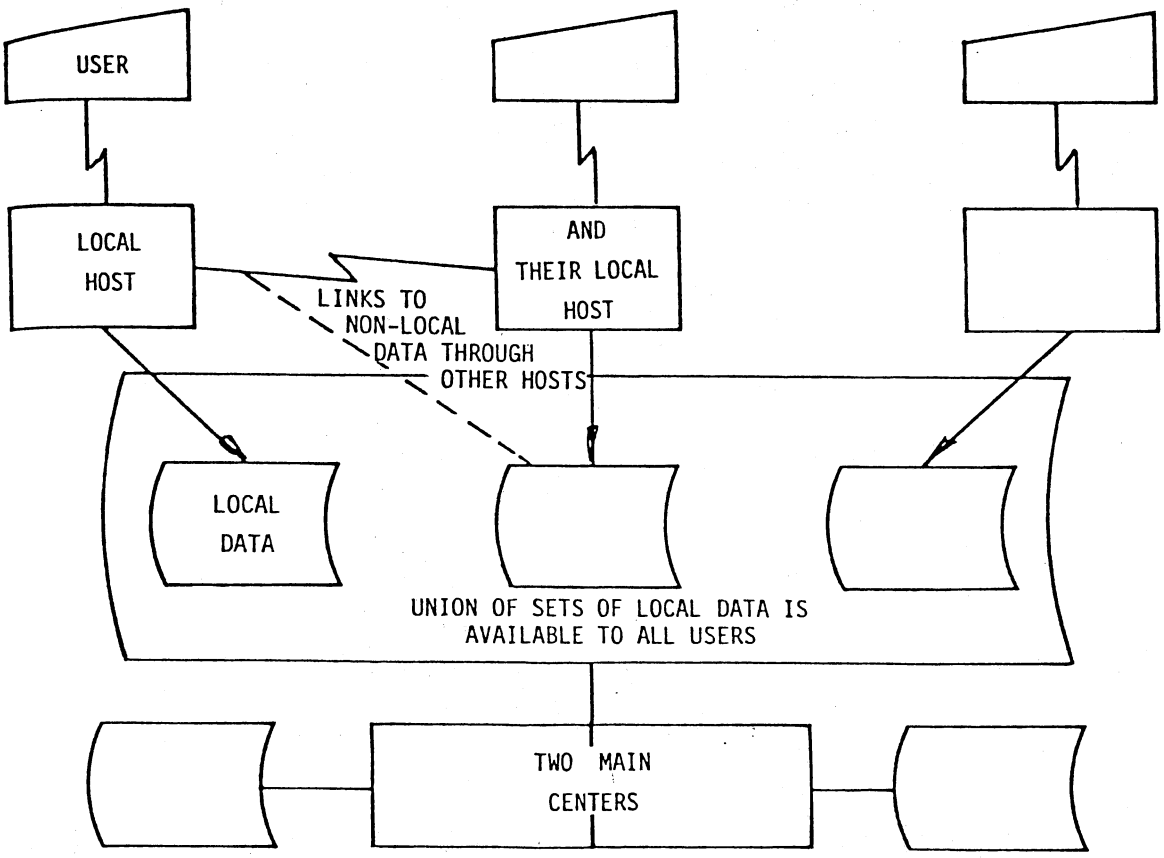
The U.S. Geological Survey uses a DBMS for its Ground Water Site Inventory System (6). The System consists of approximately 760,000 records and requires 500 megabytes of disk storage. USGS state field offices interactively update the database via telecommunications about 50,000 times a week. During the next 5 years, the USGS plans to convert to a new DBMS which would employ minicomputers in the state offices to process some of the data before transmitting it to the Washington D.C. office.

Although most DBMS are used on large mainframe computers, the relatively recent availability of low cost processing in the form of minicomputers and microcomputers has created a new trend--the distributed DBMS. In the distributed DBMS, each user can have a local processor which can communicate with other local processors. In addition, the local processors can communicate with a main processor. As an example, consider Figure 1 which depicts the distributed database used by the Bank of America in California (7). Processors are spread throughout California. If a processor receives a transaction of an account located in another processor's database, the transaction is forwarded to that database without involving the main processor. The main processor is then sent a record of all processing.

Types of Database Management Systems

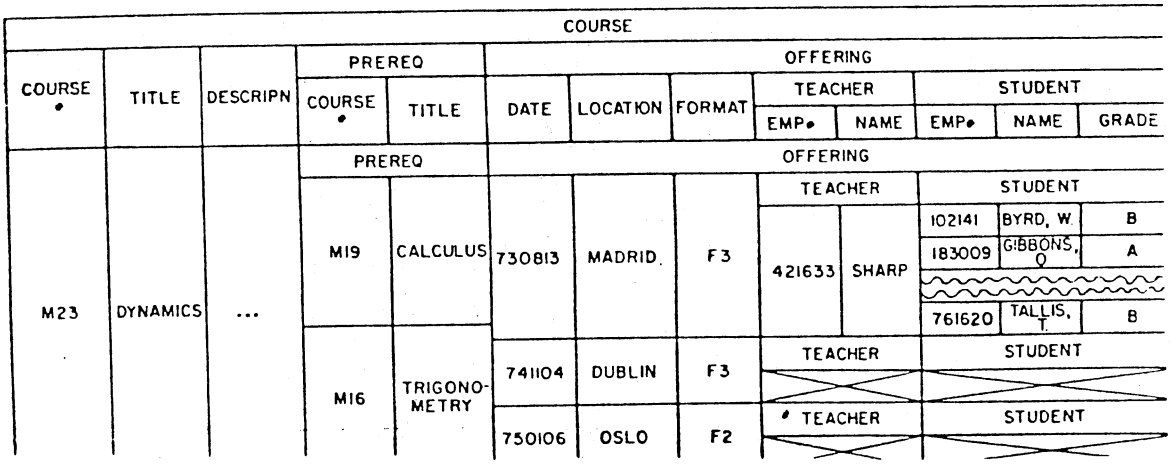
There are three basic types of DBMS: hierarchical, network, and relational.

A hierarchical DBMS might be used by a university to keep track of enrollment. This DBMS is depicted in Figure 2 (8). As the name implies, the database consists of a hierarchy of names, each name being composed of one or more subnames. In this case, the highest ordered name is course. Course consists of the course number, title, description, list of prerequisites, and a list of course offerings. Each prerequisite is made of a course number and title. Each offering has a date, location, format, teacher, and student.



Source: Zemrowski (7).

Figure 1. A Distributed Database



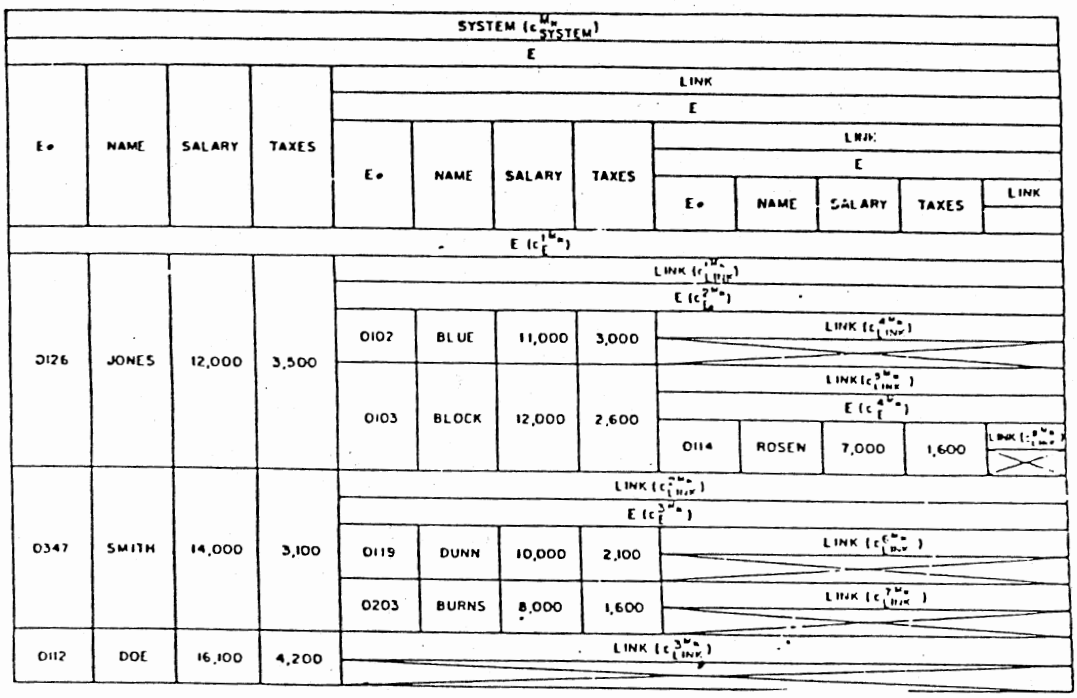
Source: Jacobs (8).

Figure 2. Hierarchical Database

Each teacher and student is then divided into a number and name, with the addition of a grade for the student. Thus, in a hierarchical database, the same record type may not appear above and below itself. In addition, every record type can have at the most, one immediate parent. Although this database appears to be designed primarily for enrollment, it could also be accessed by a program to calculate student grade point averages.

Related to the hierarchical database is the network database, demonstrated in Figure 3 (8). In this example, the external or highest order name is the system, which consists of a single E table which in turn consists of an employee number, name, salary, taxes and E-Link table. Within the E table, the same structure is repeated. Thus, in a network database all the records are of the same type. The semantics of this example is that supervisors are represented by rows and those they supervise are in the E table embedded in the link table component of their row.

An example of a relational database is shown in Figure 4 (8). The relational database is a collection of separate data structures which are combined, or related by another data structure. The structure named SP relates the structure S and the structure P. So, if saleman Smith sells 400 blue screws manufactured in Rome, only a single row of SP is required to record this transaction, namely the row where S#=S1, P#=P3, and QTY=400.



Source: Zemrowski (7).

Figure 3. A Network Database

S			
S#	SNAME	STATUS	SCITY
S1	SMITH	20	LONDON
S2	JONES	10	PARIS
S3	BLAKE	30	PARIS
S4	CLARK	20	LONDON
S5	ADAMS	30	ATHENS

SP		
S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

P				
P#	PNAME	COLOR	WGT	PCITY
P1	NUT	RED	12	LONDON
P2	BOLT	GREEN	17	PARIS
P3	SCREW	BLUE	17	ROME
P4	SCREW	RED	14	LONDON
P5	CAM	BLUE	12	PARIS
P6	COT	RED	19	LONDON

Source: Zemrowski (7).

Figure 4. A Relational Database

Database Management Systems For Microcomputers

Within the past few years, microcomputers and microcomputer storage devices have been drastically reduced in price. As a consequence, more people are using microcomputers for database management processing. Since it would be an excessively long task to report on all the DBMS for microcomputers, 2 of the more popular microcomputers, the Apple and the TRS-80, will be used in a discussion of DBMS for microcomputers.

The Apple computer uses a pseudo-hierarchical DBMS called DB Master (9). Each database is a single file. Each file consists of records which use the same form. In other words, the forms within each record are similar (asks the same questions). The form consists of information called fields, a screen full of information being called a page. There are 7 different types of fields in DB Master. Of these, 5 are numeric and 2 are character. The numeric fields have the following ranges: 1) 0-255, 2) +/- 32767 3) +/- 999,999,999. One numeric field handles only dollars and cents, another is called a computed field, being some algebraic manipulation of the other numeric fields. Of the 2 character fields, one can store up to 30 characters, the other handles only the 'Y' or 'N' characters. By specializing the purpose of these fields, DB Master is able to more efficiently manipulate and store information. If the system were truly hierarchical, a record of one file would be able

to access information within a record of another file. Such intrafile access is too complex for most microcomputer software at present--something that will be overcome in the not too distant future. In any event, a record can have up to 10 fields (or 35 bytes, whichever comes first) to be defined as key fields. The key fields are used to recall data. The user can perform different searches based on how the key fields are defined. Fields can be defined four different ways. First, a range of values for a particular field can be defined. In addition, all of the relational operators (less than, greater than, etc.) can be used to define a search for particular data. Also, 2 or more different key fields can be combined in a logical 'and' or logical 'or'. Finally, each field can search for a wild card character. For example, if a search is to be performed for a particular value of a field but the spelling of that value is not certain, then a wildcard search character can be placed in the position of the uncertain letter. The number of records per file depends on the record length, the number of fields per record, and the number of key fields. At most, the DB Master could contain 387,045 records per file, at least, about 76 records per file.

The TRS-80's DBMS is similar to DB Master and is called Profile (10). Profile creates 4 files which are called segments of a single database name. Each segment can contain up to 36 fields, where the segment must not exceed 255 characters in total. Profile distinguishes between different field

types for correct keyboard entry. Basically, there are about 5 different fields: alphanumeric, numeric, dollar-cent numeric, computed, and date. The computed field is defined as some algebraic manipulation of other numeric fields. Profile actually has many other field types, but they are all variations of the groups mentioned. Profile can perform all of the searches described for DB Master. The key fields are within segment 1.

To summarize, most of the DBMS for microcomputers are related to a hierarchical type DBMS. Information is stored in fields and usually catagorized to one or two levels. Some of the fields are defined to be special sort fields which are used to generate a particular subset of the database. All of the relational operators can be used to define an acceptable range for the special sort fields.

CHAPTER IV

SYSTEM DESCRIPTION

Aim 65

The Aim 65 microcomputer is manufactured by Rockwell as a general purpose microcomputer based on the 6502 microprocessor (see Figure 5). The Aim's purpose in the data management system is to read data from cassette tape and send it to the Tektronix 4052. To do this, the TTY interface on the Aim was altered to function as a RS-232 interface.

One of the functions in the Aim's monitor is the execution of machine language code begining with location B000 whenever the '5' key is depressed. By locating a program at B000 on the Aim, another computer need only send a '5' character to start the program.

Tektronix 4052

The Tektronix (Tek) 4052 microcomputer (see Figure 6) is capable of executing a graphics version of BASIC programming language. The 4052 uses 2 Motorola 6800 processors, one to handle I/O and one to handle instructions. The 4052 has both a general purpose interface bus (GPIB) and an RS-232 interface.

The 4052 is the center of the data management system.

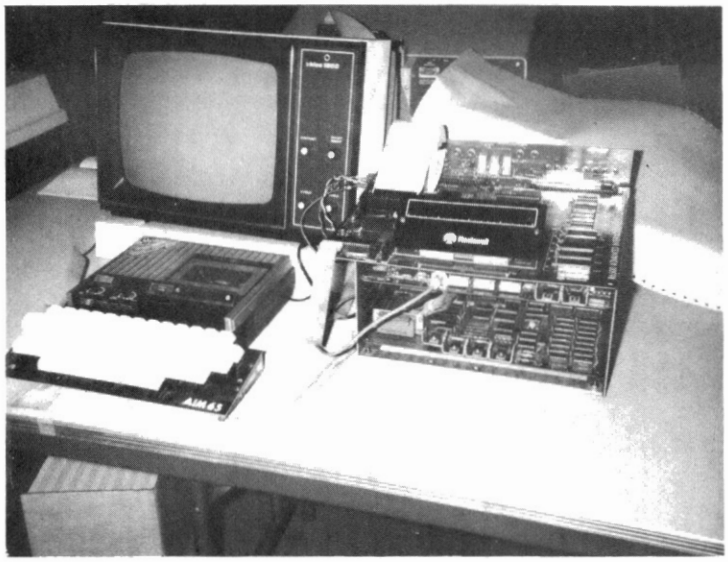


Figure 5. Aim 65 Microcomputer

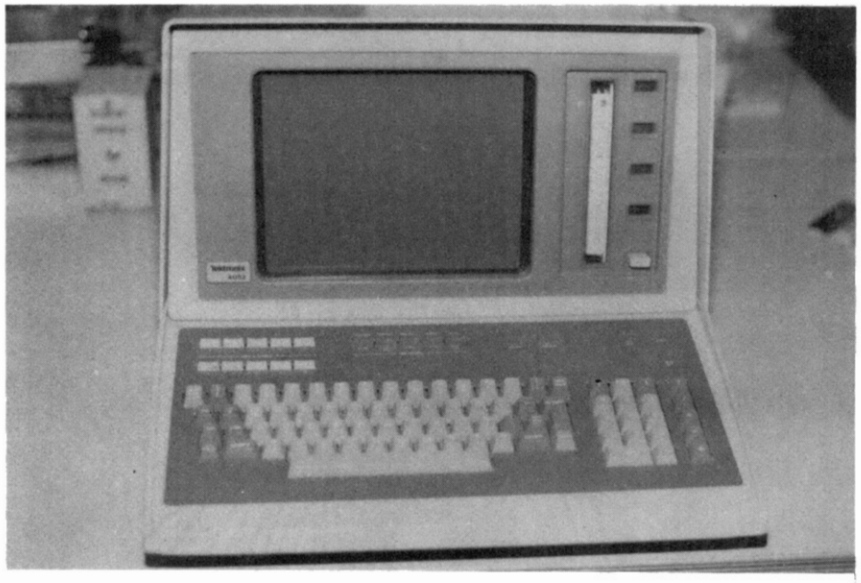


Figure 6. Textronix 4052

In fact, the programs which perform most of the functions of the system operate on the 4052. In addition, the data is stored on 4052 cartridge tapes. The 4052 can also serve as a graphics terminal, accepting graphics commands from the University Computer or a BASIC language program.

The screen is a long life phosphorous tube which automatically dims after 90 seconds of keyboard inactivity. There are 150 horizontal and 100 vertical graphics positons on the screen.

There are 54K bytes of RAM available for user programs on the 4052. To store data, the 4052 uses magnetic tape cartridges capable of recording approximately 300 kilobytes. String variables can have only one dimension, with 26 possible names (A$-Z$). There are 260 numeric variable names capable of multi-dimensioning (A0-A9,B0-B9,etc.). The 4052 has an input buffer which can accept up to 255 characters independent of the execution of a BASIC program. This feature is used to allow the Aim to transfer an experimental unit at the same time the user is recording the previous experimental unit.

Tektronix 4662

The 4662 digital plotter (see Figure 7) is capable of plotting graphs generated with the BASIC graphics commands or graphics commands sent by the University Computer. The 4662 has both a GPIB and a RS-232 interface. The plotter has an internal buffer capable of storing graphics commands.

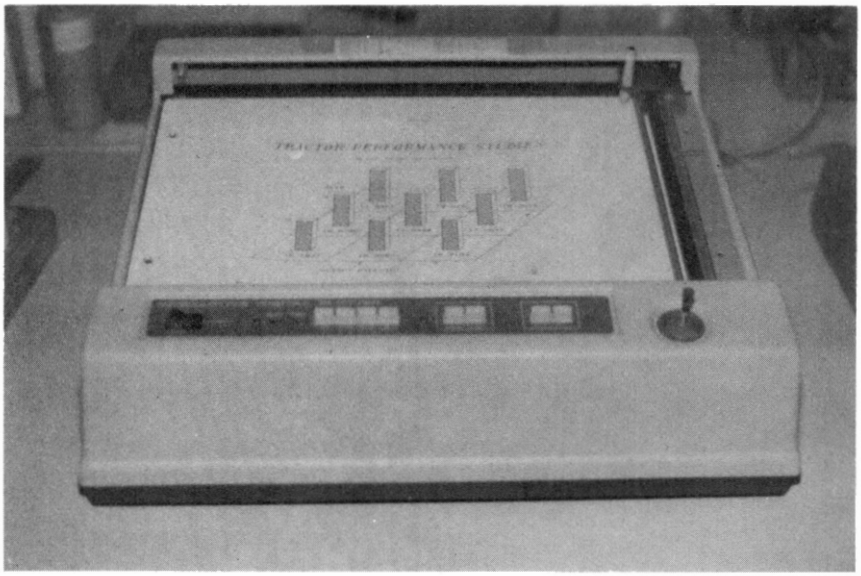


Figure 7. Tektronix 4662

Texas Instruments 8010 Printer

The 8010 printer (see Figure 8) is a dot matrix printer used to obtain a printed copy of a dataset stored on the 4052's cartridge tapes.

General Operation

The data management system is composed of 3 programs which operate on the 4052: The Aim to Tek program, The Editor, and The Tek to IBM program. Each of these programs make use of the user definable keys located in the upper left hand corner of the keyboard (see Figure 9). When a user definable key is pressed, it causes the 4052 to begin executing at a unique line number for each key. The keys act like an interrupt, returning the 4052 to the previous state of execution when the key routine encounters a return statement.

The Aim to Tek program has 7 functions which allow experimental units to be transferred from cassette tape to 4052's cartridge tape while adding descriptive information to the experimental units. A complete listing of the Aim to Tek program is found in Appendix A. The 7 functions correspond to the 7 user definable keys which are indentified by a template laid over the keys (see Figure 10). Descriptive information which is automatically added by the 4052 to each experimental unit is called a file parameter. Two keys allow the operator to add or delete file parameters. A block parameter is descriptive information which is added by the

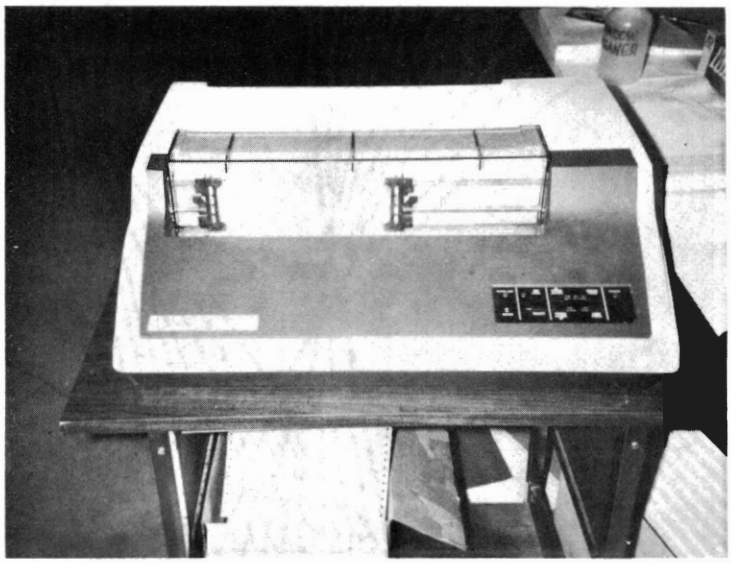


Figure 8. Texas Instruments Printer

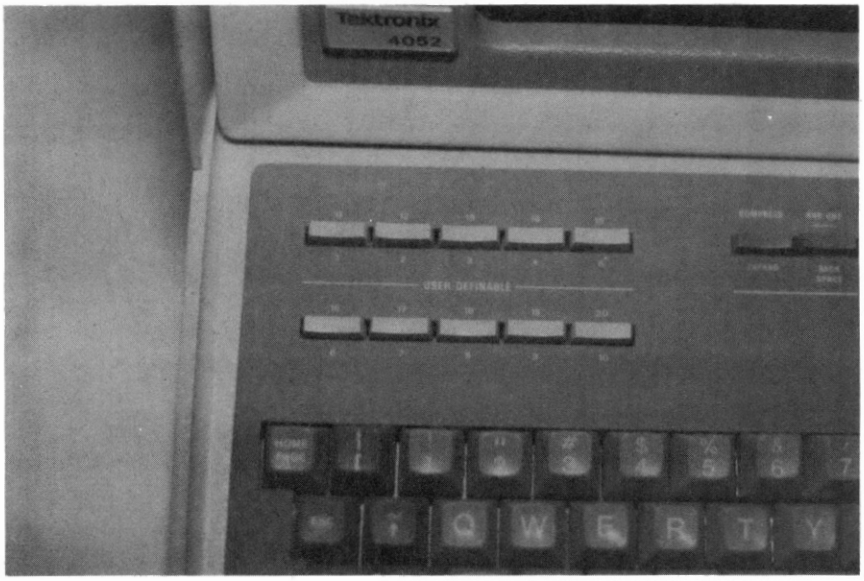


Figure 9. User Definable Keys

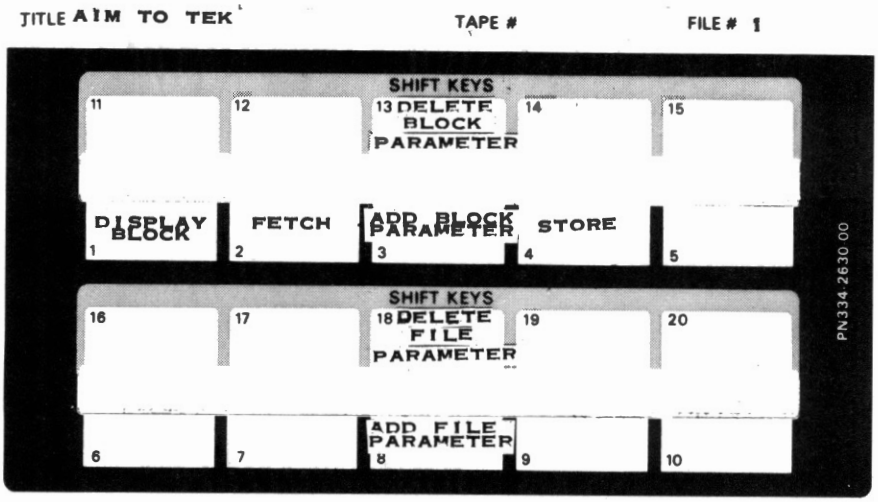


Figure 10. Aim to Tek Template

user to the experimental unit currently in memory. As with file parameters, two keys allow the user to add or delete block parameters. By far the most complex function of the Aim to Tek program is the fetch routine. When the fetch key is pressed, the 4052 waits for the input buffer to be filled with an experimental unit. When an experimental unit is received, the 4052 transfers the contents of the input buffer into RAM and then 'tells' the Aim to send the next experimental unit. While the Aim is busy doing this, the 4052 converts the newly received experimental unit from ASCII characters representing hexadecimal data to numeric decimal data. After applying calibration constants to obtain numbers in engineering units, the experimental unit is converted back to ASCII characters representing decimal data. The last operation is the addition of any previously defined file parameters. The display function prints on the screen the experimental unit currently in memory along with any parameter information. The store function records the experimental unit in memory on magnetic tape.

The Editor program has 16 functions which allow the ASCII data files to be altered and interactively transferred to or from the host computer, an IBM 370. A complete listing of the Editor program is found in Appendix B. The Editor can operate on any ASCII data file less than 45 kilobytes and with a maximum of 132 character records separated by a carriage return. Thus, it can be used for text and programs, as well as experimental data recorded by the Aim to Tek pro-

gram. As with the Aim to Tek program, a template is placed over the user definable keys which defines each of the 16 functions (see Figure 11). The recall key inputs an ASCII data file from memory while the store key performs the complimentary function. All the data is contained within one character string dimensioned to 45000. Within this string, begining and ending positions identify the current line (maximum length of 132 characters). Moreover, all of the editing functions are line oriented. The line key prints the current line, the line up key defines the current line to be the previous line and prints it, and the line down key is similar to the line up key only in the opposite direction. Likewise, there are 3 page keys which work the same way, a page being 20 lines. A special page key, page 1, defines the current line to be the first line of text and then prints 20 lines from there. The delete key erases the current line from text, while the insert key allows lines to be inserted just prior to the current line of text. The add key concatenates lines to the end of text. To alter characters within a line, the find/replace key can be used. The print file key will generate a printed copy of a file on the printer. The send and receive keys allow the user to transfer files to or from the host computer connected to port 40. At present, these keys are set to operate on TSO with an IBM 370 as the host.

The Tek to IBM program has 11 functions which allow the operator to display, print, or transfer (to the IBM 370) a

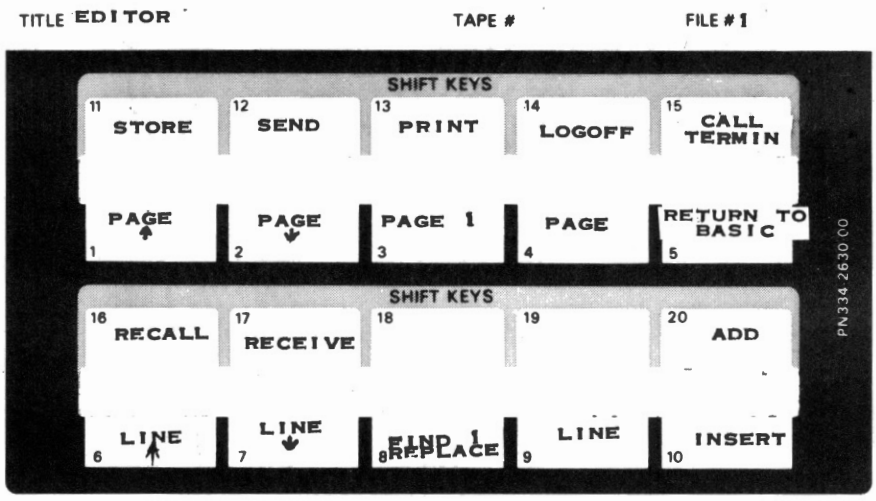


Figure 11. Editor Template

selectable subset of one or more experimental data files. A complete listing of the Tek to IBM program is found in Appendix G. As with the other 2 programs, a template is used to associate the user definable keys with the 11 functions (see Figure 12). Each of the 5 keys: pull, speed, fuel, slip, and rpm can be used to prevent a particular variable from being transferred or to set a particular range on a variable to be transferred. For example, pressing the pull key for the first time will cause the computer to respond with a message indicating pull will not be transferred. If the pull key is pressed again, the computer will ask for minimum and maximum values acceptable for pull. Each of the 5 variables has a key which toggles between these two functions. In addition, the add search parameter and the delete search parameter keys can be used to identify particular descriptive information associated with the data to be transferred. For example, if only data obtained from a field cultivator is desired, then the search parameter information should be 'field cultivator' (or whatever characters have been used to identify field cultivator).

The display search parameter key prints on the screen both the variables to be transferred and the parameter information to be used in the search. After identifying the information on which the search is to be based, the actual search begins when the search print, search display, or search transfer keys are depressed. The only difference in these 3 keys is where the output is sent. The search print

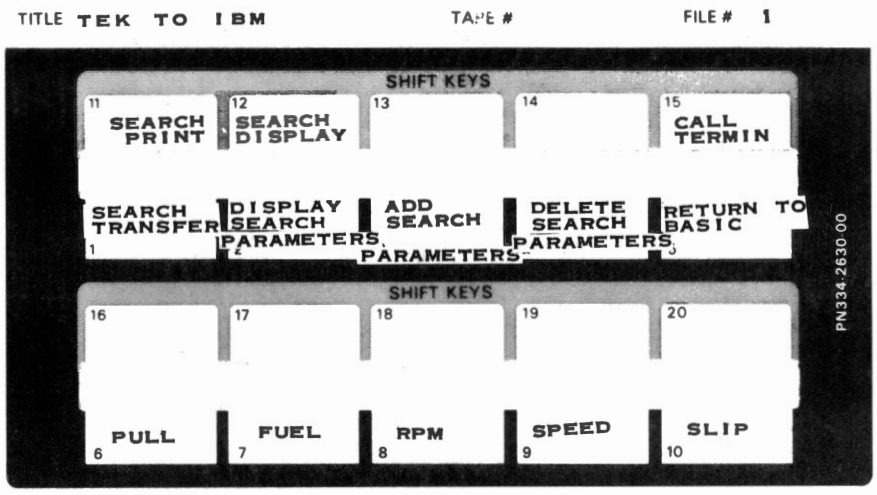


Figure 12. Tek to IBM Template

displays the output on the line printer, the search display prints the output on the screen and the search transfer prints the output over port 40 to the host computer.

CHAPTER V

PROGRAM DESCRIPTION

Aim to Tektronix

Add File Parameters (8030-8110)

The add file parameter routine concatenates (see Figure 13) keyed-in character information onto the file parameter string, P$. The file parameters are automatically added as block parameters during the fetch routine. The first two lines (lines 8030-8040) allow the user to enter the file parameters. If a null string is entered, the routine ends (line 8045). If characters are entered, the program concatenates a space in order to visually separate the parameters (line 8050). The proposed concatenation is examined to see if the new line will fit within 72 characters (line 8060). Within any line, the first 30 characters hold the decimal data so the maximum length of P$ is 42. But the operator should beware that using 42 characters for file parameters leaves none for block parameters. If the file parameter will be too long, the user is notified (line 8100) and the routine ends (line 8110). If the additional file parameter (stored in A$) will fit, it is concatenated onto P$ (line 8070), and the routine ends (line 8090).

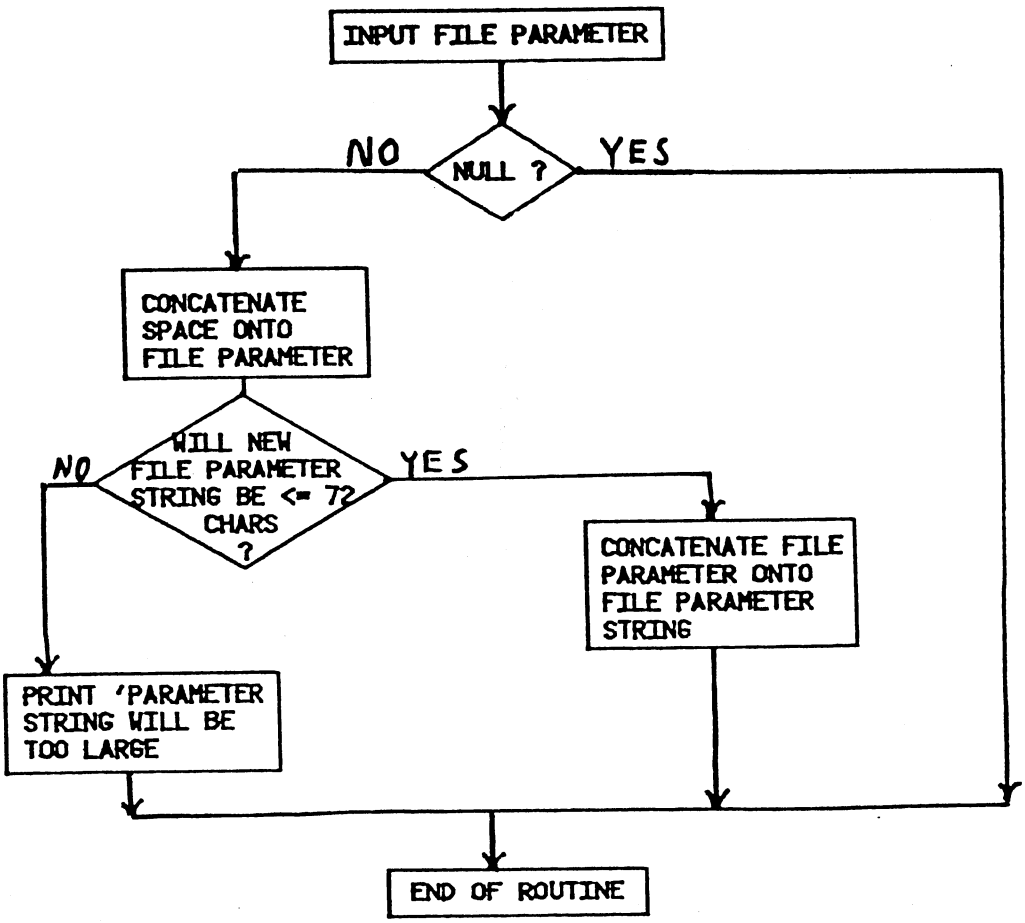


Figure 13. Flowchart of Add File Parameter Routine

Add Block Parameters (5000-5150)

The add block parameter routine concatenates keyed-in character information onto the end of each line in the decimal data array, D$ (see Figure 14). The first 4 lines are similar to the begining of the add file parameter routine. Then the decimal data array, D$, is searched from the first character position (line 5070) to the first carriage return (line 5080). Assuming there is room for the concatenation (line 5090), the keyed-in information is inserted just before the carriage return, and the position to begin the search is redefined for the next line (line 5110). This procedure is repeated for all ten lines of D$ (line 5120). When the starting position, X, exceeds the length of D$, the routine ends by jumping to the display block routine (line 6120).

Delete File Parameters (7030-7080)

The delete file parameter routine removes file parameters from the file parameter string, P$. The first 2 lines allow the user to enter the character string to be deleted. The third line determines if these characters exist within the file parameter string. If they don't, the routine ends by displaying the unchanged file parameters (line 7060). Otherwise, the file parameter and the space which follow it are deleted from P$, and the new P$ is displayed (lines 7070-7080).

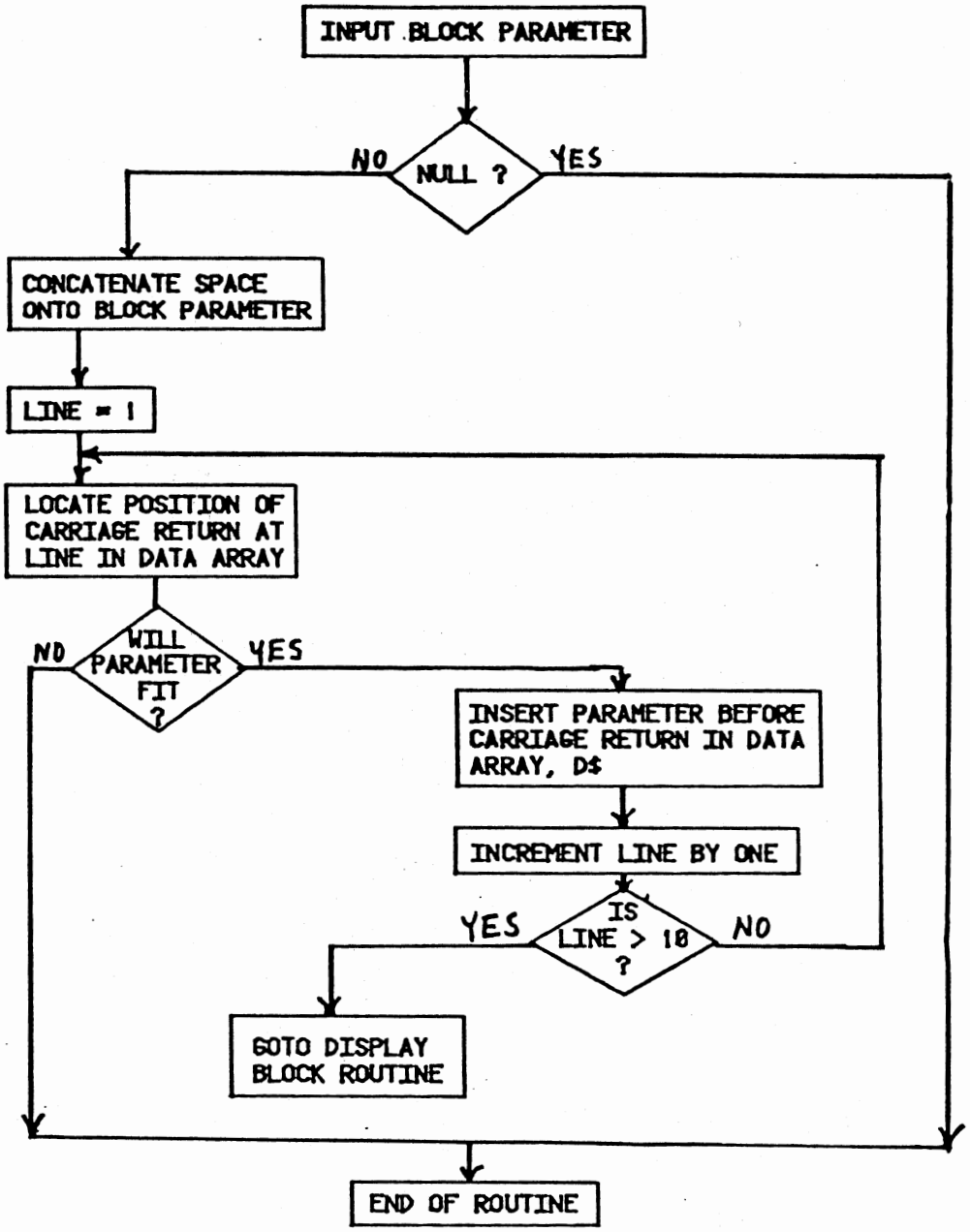


Figure 14. Flowchart of Add Block Parameter Routine

Delete Block Parameters (6030-6120)

The delete block parameter routine allows the user to remove block parameters, including previous file parameters that may have been automatically added (see Figure 15). Additional steps are required to ensure only file parameter and block parameter characters are altered. After the user enters the characters to be deleted (lines 6030-6040), the number of characters in the first line of D$ is determined (line 6050). Then the first line of D$ is searched begining with the 31 st character position since the first 30 characters hold the decimal data (line 6060). If these characters are not found, then the deletion is not performed (line 6070). If they are found, the deletion is performed over all ten lines of D$ (lines 6080-6110). The routine ends by jumping to the display block routine (line 6120).

Display Block (3040-3070)

The display block routine (see Figure 16) clears the screen (line 3040), prints column headings (line 3050), and prints the decimal data array along with any parameter information (line 3060). The maximum length of the decimal data array is 720, in which case there would be 10 lines of 72 characters. By default, the 4052 automatically issues a line feed whenever a carriage return is printed on the screen.

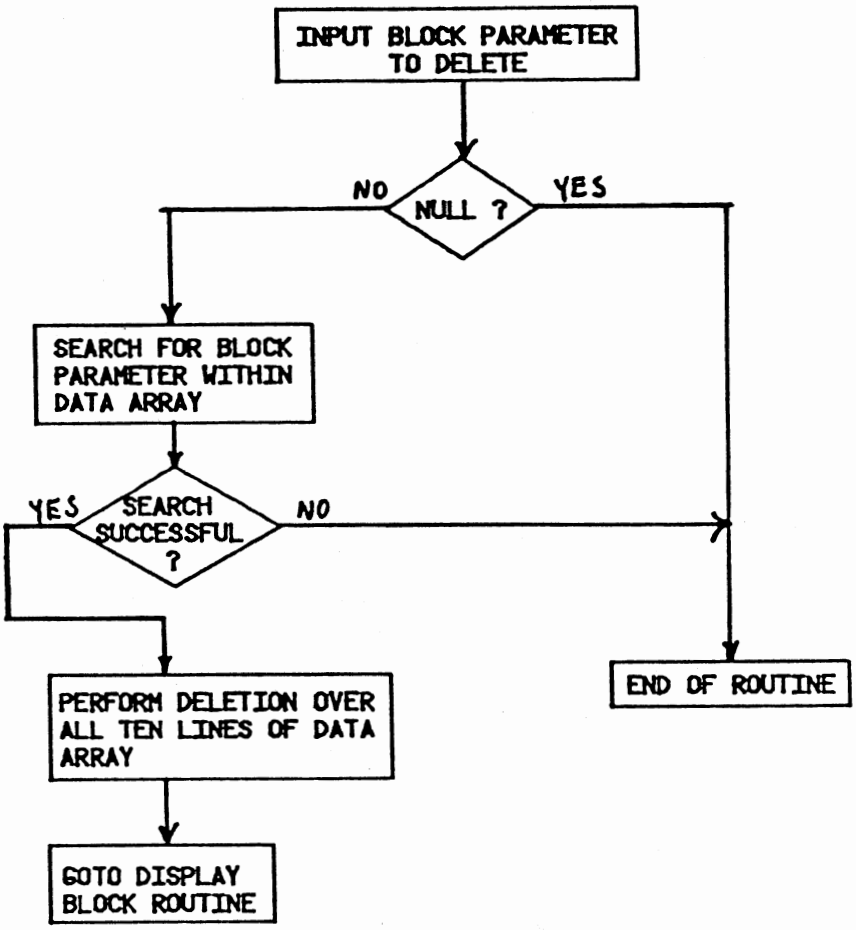


Figure 15. Flowchart of Delete Block Parameter Routine

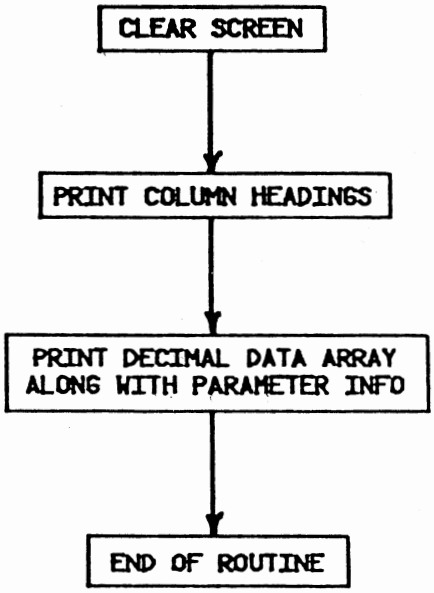


Figure 16. Flowchart of Display Block Routine

Store Block (4030-4170)

The store block routine records the decimal data array, D$, on magnetic tape (see Figure 17). The first line (line 4030) checks the value of T9, which indicates if the block to be recorded is the first block of the file. If it is, the user is given the chance to locate a particular file (lines 4040-4070). If T9 is not zero, then the user is asked if this will be the last block recorded (line 4080). In any event, the block is recorded. If it's the last block, the end of file character (EOF) is also recorded (line 4120), and the program re-initializes (line 4120).

Fetch (2000-2960)

The fetch routine inputs an ASCII character string transferred by the AIM 65 microcomputer (B$), and mathematically manipulates it into another ASCII character string called the decimal data array (D$) (see Figure 18). First, the 2 strings B$ and D$ are cleared. If this is the first block being fetched (indicated by S9) since the program was initialized (line 2050), the internal input buffer is cleared (line 2090), and the Aim 65 program is activated (line 2100). Clearing the input buffer enables the 4052 to accept up to 255 characters previous to, and during any subsequent input operations. If it's not the first fetch, the routine jumps straight to a dwell where the 4052 waits for the input buffer to be filled by the Aim (lines 2110,2120). In most cases, the 4052 discovers the input buffer is

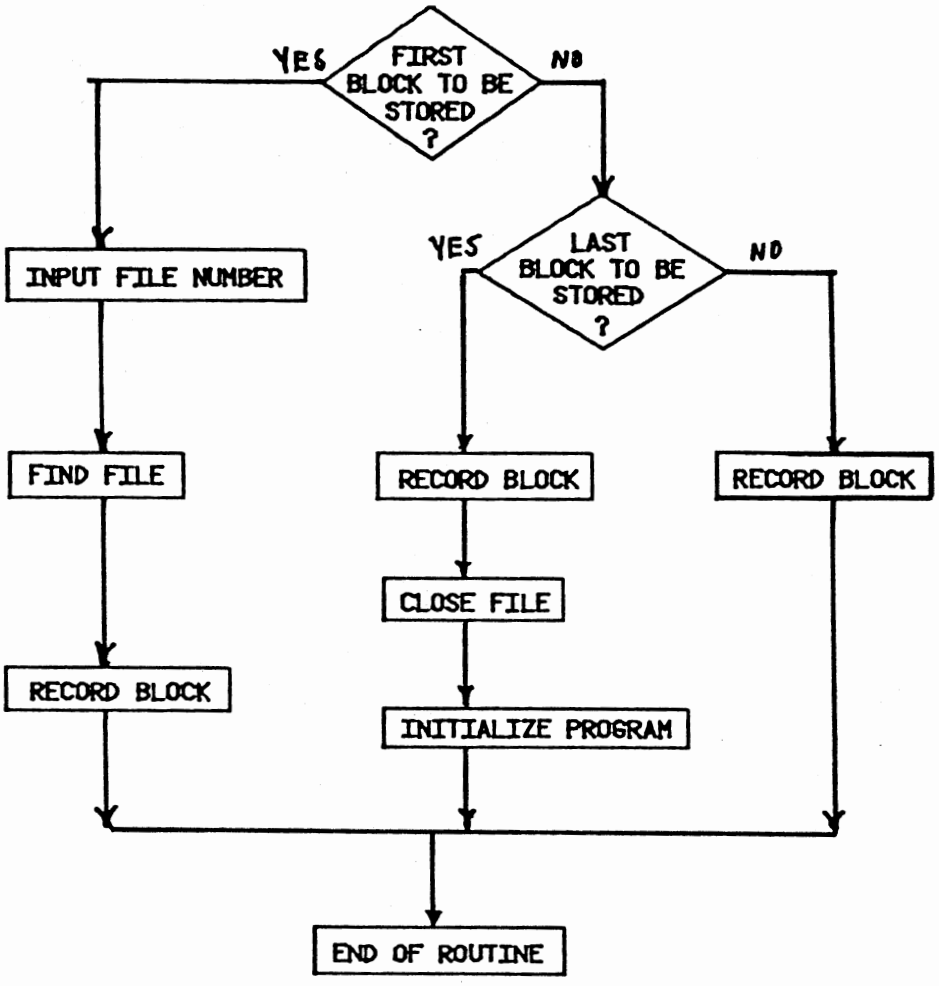


Figure 17. Flowchart of Store Routine

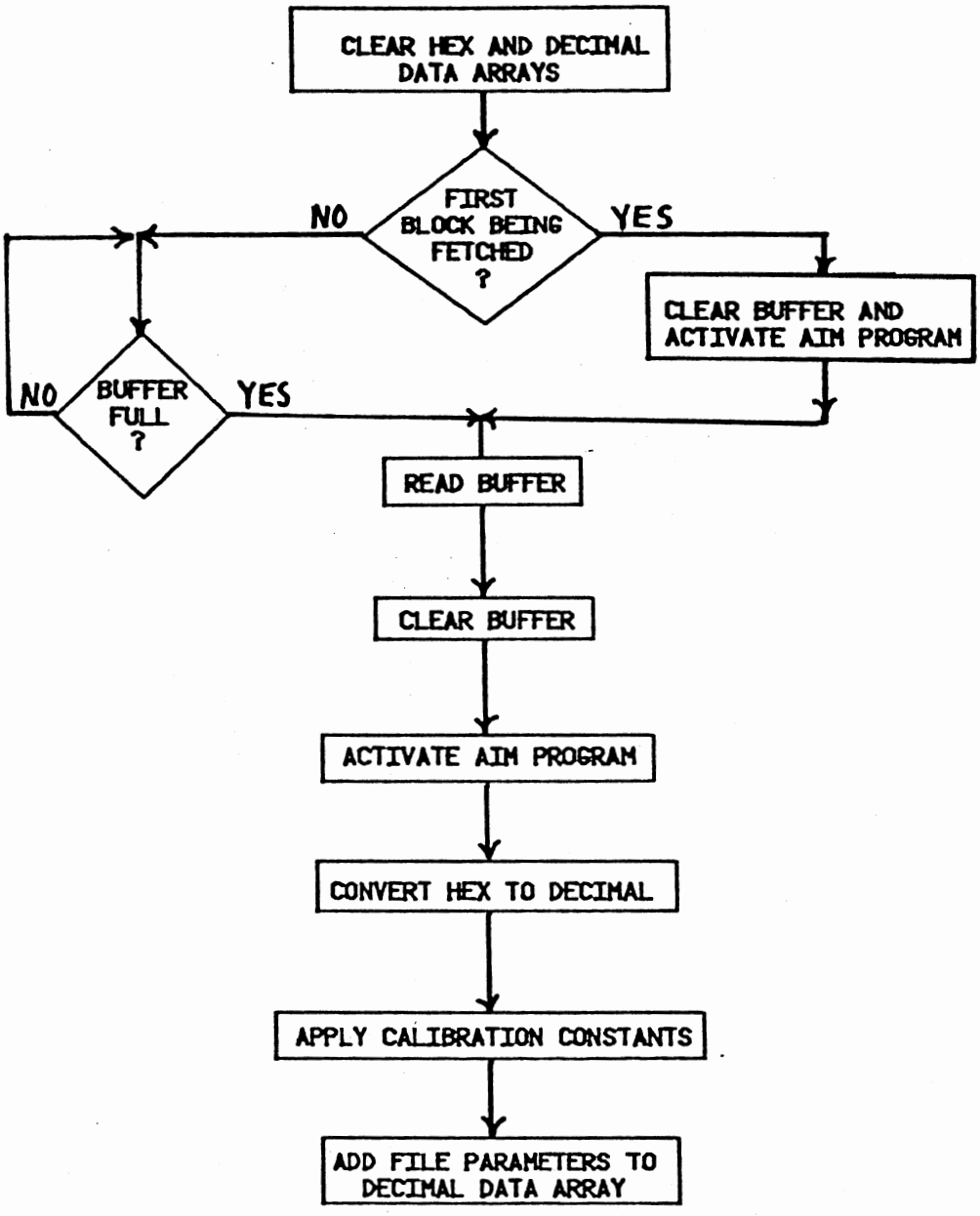


Figure 18. Flowchart of Fetch Routine

already full. This is because the last fetch reactivated the Aim 65 program after storing the contents of the last input buffer in the string variable, B$. If the Aim 65 was successful in reading the cassette and transferring the data, the input buffer will have 234 characters in it. These characters correspond to 117 bytes in the Aim 65 microcomputer (see Figure 19). If the Aim was not successful, the hex string, B$, is cleared and the routine ends (lines 2180-2210). Assuming a valid hexadecimal string has been accepted and stored in B$ (line 2160), the program reactivates the Aim program (line 2240). So while the 4052 is busy executing the rest of the fetch routine, the Aim will be transferring the next experimental unit to the 4052's input buffer. Some of the locations in B$ have numbers stored in low-high byte order and thus need interchanging (lines 2280,2290). The actual conversion is performed by a subroutine (line 2310). Lines 2270 through 2340 convert quantities corresponding to pull, T(1-10), fuel flow, T(11-20), rpm, T(21-30), ground speed, T(31-40), and wheel speed, T(41-50). Lines 2350-2400 convert the calibration constants, T(51-58), which are already in high-low byte order and thus do not need interchanging. One calibration constant, T(58), requires 3 bytes (lines 2410-2430).

The next section, (lines 2550-2690) performs the mathematical operations to generate numbers with appropriate engineering units. Should a divide by zero occur, a subroutine sets that particular quantity to 9999 (line 2570) and

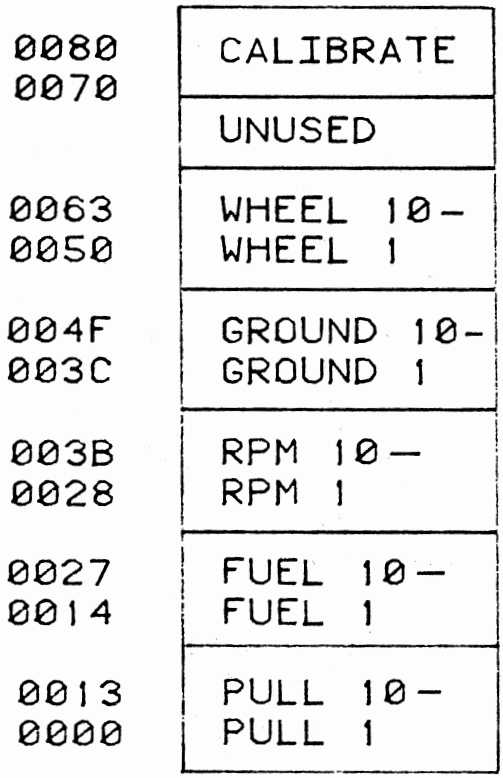


Figure 19. Memory Map of Aim 65 Data

normal execution continues. Wheel and ground speed values are combined to generate percent wheel slip, which redefines wheel speed (line 2650,2660). The integer function is used throughout this section to eliminate insignificant digits (the integer function does not round off, simply truncates). The next group of lines (lines 2810-2930) converts numbers in array T into a string array, D$, which when displayed, is a 10 by 5 matrix with the 5 variables, pull, speed, fuel, slip, and rpm as columns. Within each observation, each variable is converted into a string (line 2850), and then buffered with spaces until the string has 6 characters (lines 2860-2880). This improves the appearance of the data. The 6 characters are then concatenated onto the end of A$ and the process repeats for the next number within the observation (lines 2890- 2900). When all 5 numbers have been added to A$, A$ is concatenated onto D$, the decimal data array. Since this is the end of a line, a carriage return is also concatenated (line 2920). The routine then repeats this procedure on the next 5 variables, which will make up the second line of D$ (line 2930). When D$ is complete, containing 10 lines with 30 characters per line, the fetch routine jumps to the add block parameter routine where the previously defined file parameters, P$, are added to the end of each line of D$. (lines 2950,2960).

Aim 65 Program (B000-B047)

The program on the Aim microcomputer reads an experimental unit from cassette and transfers it over an RS-232 cable to the 4052 (see Figure 20). A complete listing of the Aim 65 program is given in Appendix D. The first 4 lines of the program define the input device to be cassette tape number one. The first two lines identify the cassette tape, the next two identify tape number one. Then, the next 6 lines (lines B00A-B018) set the filename of the file to be read from cassette to 'AAAAA'. This filename is used by a similar program which records the data. The subroutine call to EDEA (line B01B) waits for 5 consecutive synchronization characters which readies the Aim for tape input. Then the first block is read from tape and stored in the input buffer (line B01E). The next 6 lines (lines B021-B02B) use the X index to transfer the contents of the input buffer to page zero. The input buffer can hold a maximum of 80 bytes, but one experimental unit is 117 bytes (see Figure 19). So when the first input buffer is emptied, the Aim automatically reloads the buffer with the next 80 bytes from tape. When the entire experimental unit is in page zero, the X index is used again in a similar manner to transfer the data to the 4052's input buffer (lines B02D-B037). To do this, a subroutine (JSR EA46) converts a byte in the accumulator to 2 ASCII characters (MSB first) and prints them on the output device (to the 4052). Thus, there are 234 characters which represent 117 bytes. Locations from 0064 to 006F are not

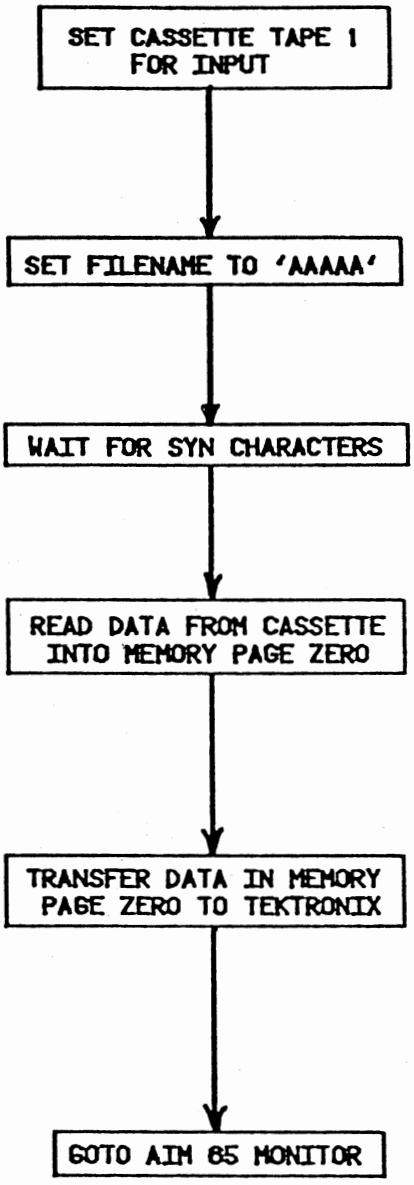


Figure 20. Flowchart of Aim 65 Program

part of the data so they are skipped (lines B039- B043). The program ends by jumping to the monitor (line B046).

Hex to Decimal Subroutine (150-280)

This subroutine converts a hexadecimal character string, H$, to a decimal numeric variable, D. X2 is a variable which indicates digit significance. X1 indicates a position in the H$. Initially, the subroutine starts at the least significant digit, in which case X1 equals the number of characters in H$ and X2 equals zero (line 160,170). Line 190 defines X to be the ASCII code number of a character in H$. The ASC function is used rather than the VAL function because the VAL would return a zero for the characters A,B,C,D,E, and F. Line 200 terminates the routine if an illegal hex digit is encountered. Line 210 subtracts 48 from X if the character is 0-9, which generates the corresponding decimal values 0-9. If the character is A-F, 55 is subtracted, which also gives the corresponding decimal value. For example, the ASC("F")=70, 70-55=15, which is the decimal equivalent of the hexadecimal digit F. In addition, the ASC("9")=57, 57-48= 9.

Activate Aim (360-400)

The Activate Aim subroutine starts a program stored at B000 on the Aim, then clears the input buffer. A '5' received by the Aim monitor program causes execution to begin at location B000. The delay (line 370) allows time

for the Aim to echo back the "5", so it can be erased from the input buffer (line 380). S9 is set to 1 indicating the first block has been fetched.

Size Interrupt (320-340)

This subroutine handles a divide by zero error generated by the mathmatical operations of the fetch routine. The quantity being solved for when the error occurred is set to 9999 (line 320) and the subroutine ends (line 340).

Editor

Print Page (100-230)

The print page routine (see Figure 21) displays 20 lines of text, starting with the current line. First, the screen is cleared and the cursor is positioned in the upper left hand corner of the screen (line 100). Then a line counter, I, is initialized (line 110) and the starting position of the current line is saved (line 112). If the current line is the last line of the text or the line counter has reached 20, the print page routine ends (lines 180-230). Otherwise, the current line, L$, is defined, based on the starting and ending character positions, C and E (line 130). Then, the line is displayed (the carriage return is part of L$, line 140). The next values of C and E are determined (line 150) and the line counter is incremented to repeat the process . When the last line is displayed (lines 180,190), the values of C and E are restored to be those of the first

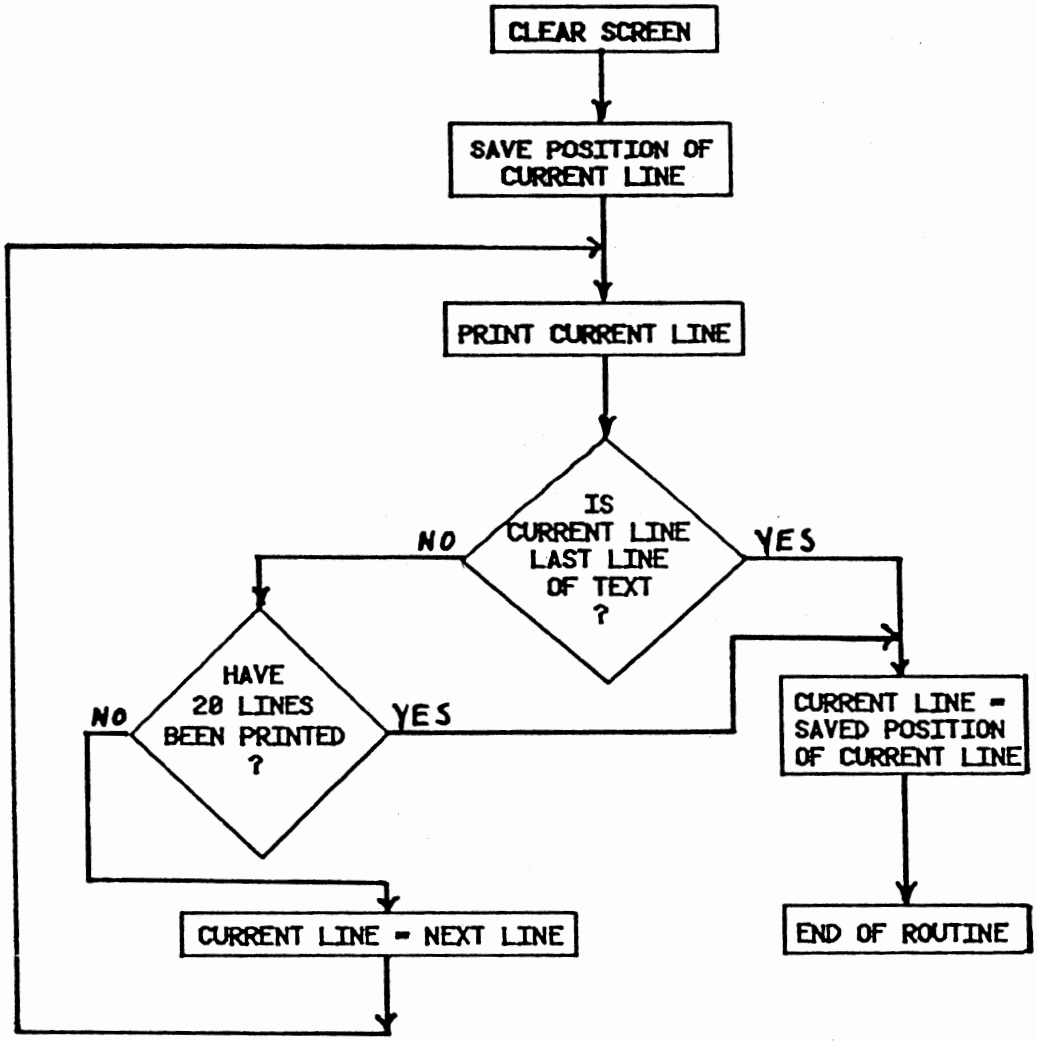


Figure 21. Flowchart of Print Page Routine

line displayed (line 200,210).

Find and Replace (510-660)

The find and replace routine (see Figure 22) finds a particular character string, replaces it with a user defined character string, and can repeat this procedure any number of times. The '/' character is used twice to separate these 3 strings. The routine locates where the separation markers are (lines 514,516) and quits if they can not be found or there aren't any characters being separated (line 518). If the format is allright, the string to find, F$, and the string to replace, R$, are partitioned (line 512). The last string, N$, represents how many times to repeat the find and replace process. If no characters were entered for N$, 1 is assumed (line 524). If there are characters indicating repeat, they are segmented out of the input string (line 526) and converted to a decimal quantity (line 528). All of the steps executed thus far only interpret the line entered by the user.

The next section actually performs the find and replace (lines 532-566). The variable, I, counts how many times the replace function (REP) is used. This is compared with, R, the number of times requested by the user, to decide when to end the routine (line 630-650). If the string to be found doesn't exist anywhere within the text, the routine ends (line 610). If it exists on a line other than the current line (line 615), the next line becomes the current line

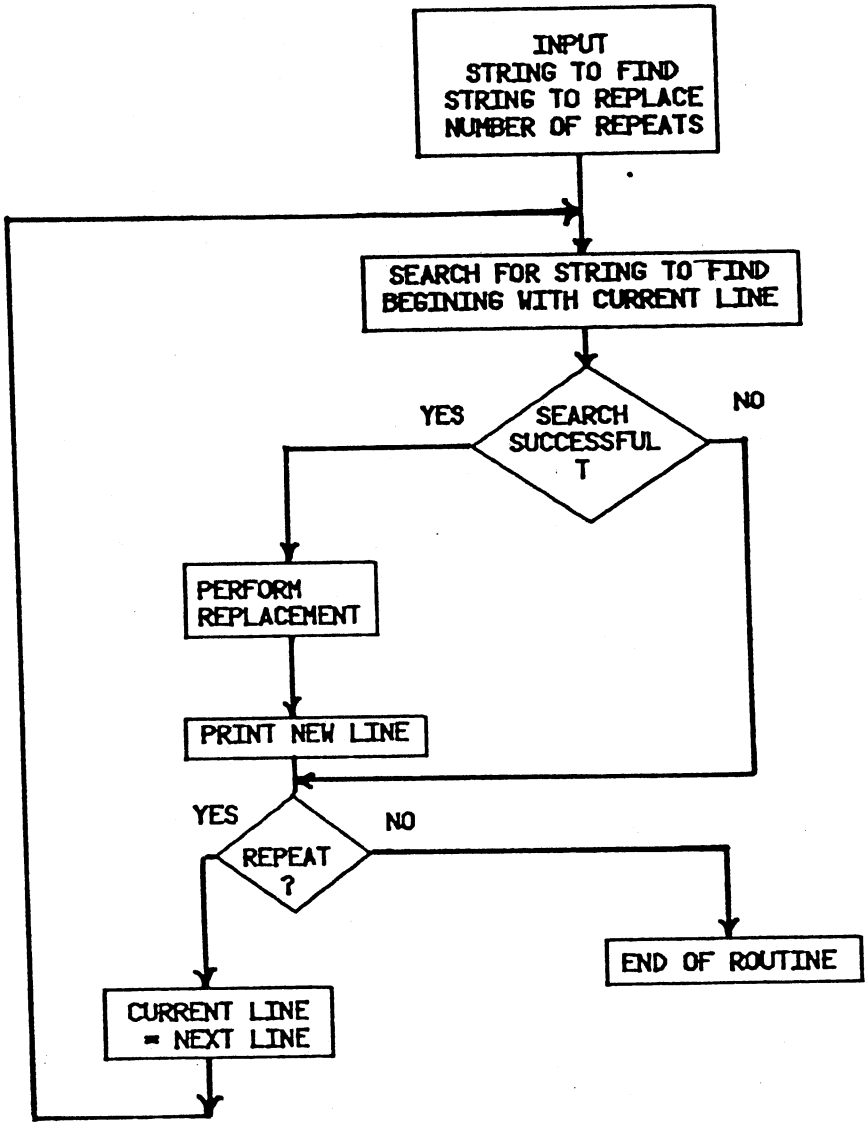


Figure 22. Flowchart of Find and Replace Routine

(line 642), and a decision is made on whether or not to repeat (line 650). On the other hand, if the replacement can be made on the current line, the counter is incremented by 1 (line 619), the replacement is made (line 620), the ending position of the current line, E, is adjusted (line 621), and the new line is defined and displayed (line 622,624). The routine then repeats for the next line if the number of repeats requested by the user, R, has not yet been obtained (line 630).

Recall Text from File (3000-3072)

The recall routine (see Figure 23) operates like the print routine except instead of a input-print cycle, an input- concatenate cycle is used. A carriage return is used to separate records (line 3042). When an end of file character is input, the routine ends by initializing the current line markers to those of the first line of text (lines 3060-3070).

Receive (3740-3920)

The receive routine (see Figure 24) accepts ASCII characters on port 40 from a host computer (in this case, an IBM 370) and records them on tape as an ASCII data file (11). It makes use of several communication routines which have been permanently recorded in 4052's memory. After locating a file (lines 3740-3742), communication parameters are set which enables the DTRECV routine to operate with an IBM 370

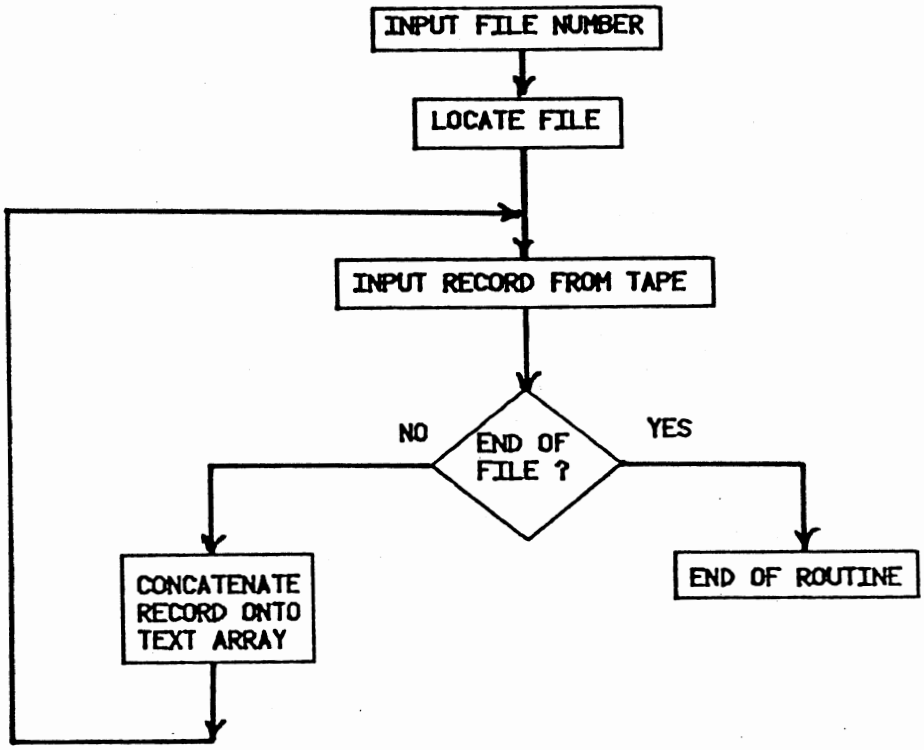


Figure 23. Flowchart of Recall Routine

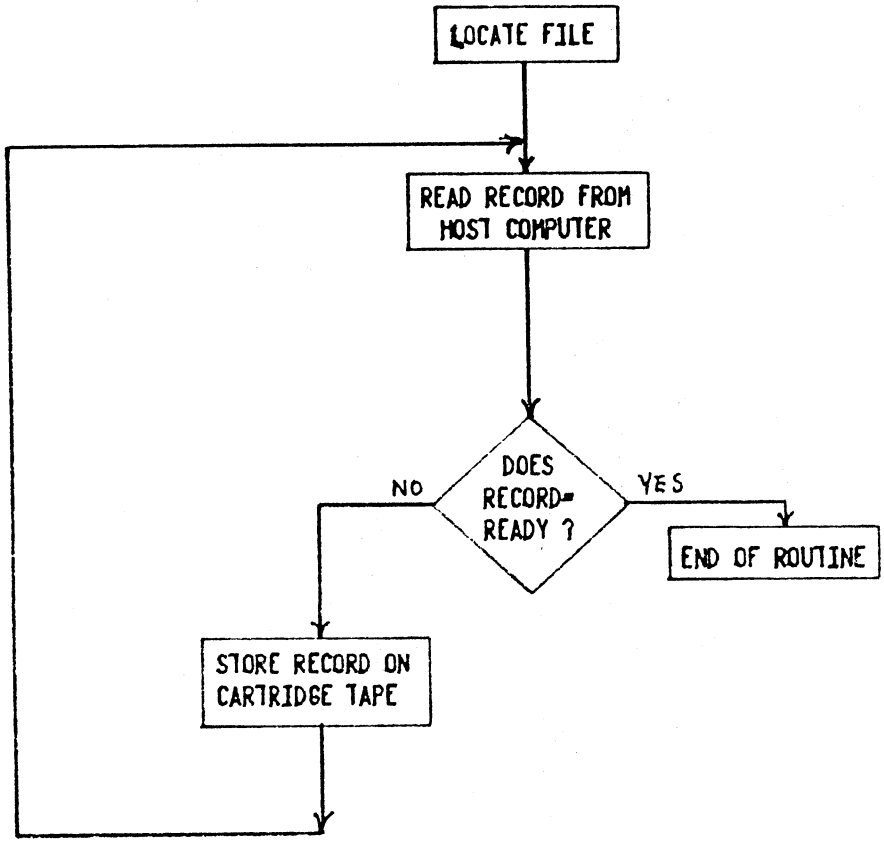


Figure 24. Flowchart of Receive Routine

(lines 3760-3840). These routines are described in detail 4052's DATA COMMUNICATION INTERFACE OPERATORS MANUAL (11). The CALL "TERMIN" statement (line 4320) begins emulation of an ASCII terminal. After the user has logged on and commanded the host computer to send a file, the DTRECV routine begins to accept characters over port 40 and records them on tape (line 3900). When the DTRECV routine receives 'READY', the end of file marker is recorded and the routine ends.

Send (4000-4360)

The send routine operates similarly to the receive routine except data is transferred from the ASCII data file to the host computer and a different set of communiction routines are used (see Figure 25) (11). As with the receive routine, the statements in this routine enable the DTSEND routine to operate with an IBM 370. Special turn-around characters used by TSO on the 370 are set so the DTSEND routine can recognize when the host is ready for another record. When an end of file marker is encountered on tape, the routine ends. This routine does not end the edit session or save the dataset on the host computer.

Initialize (4440-4490)

The Editor only operates on one file at a time, at the capacity of available memory (45K), which is the dimension of T$ (line 4450). L$ is dimensioned to 132 since the printer is capable of 132 characters per line. Both T$ and

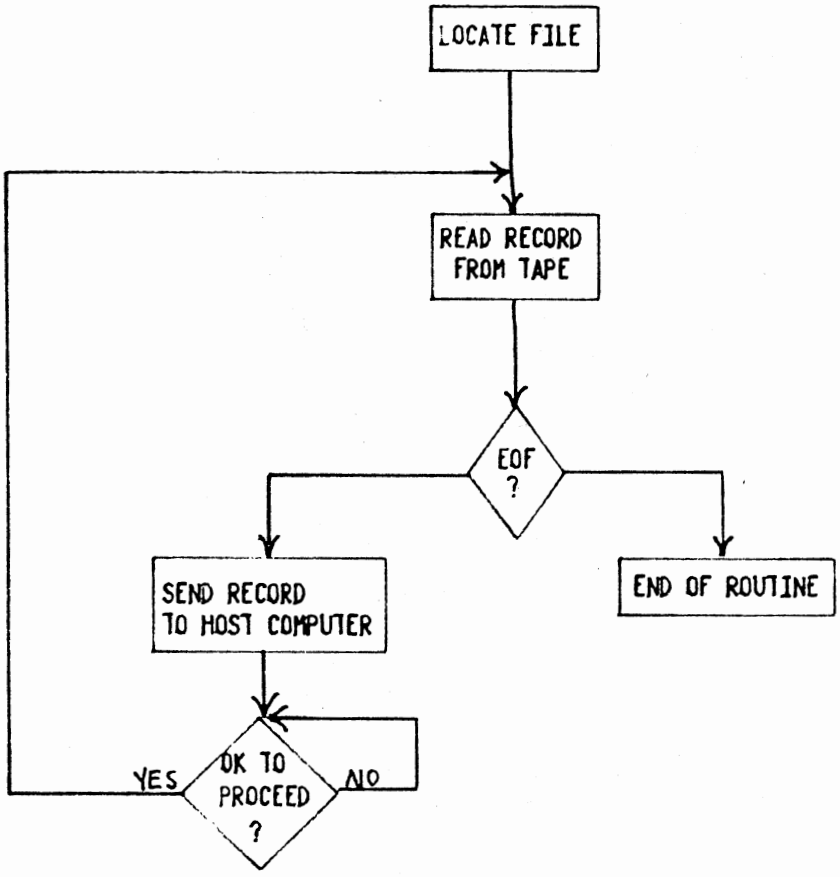


Figure 25. Flowchart of Send Routine

L$ are initialized to a null string (lines 4460,4470), M$ to a carriage return (line 4465), and the current line to the first line of text (line 4480). An internal parameter is set so whenever the screen is full, it is erased and the cursor is homed (line 4475).

Print File (2462-2506)

The print file routine prints a copy of any ASCII file on the line printer. This routine assumes a carriage return is the record separator. The user locates a particular file (lines 2462-2472), the printer interface is initialized (line 2473), and records are repetively input from tape (line 2492) and printed (line 2502). The input-print cycle is aborted when the input cycle encounters an end of file character (line 2482) (see Figure 23). If double or triple spacing is desired, the print list of line 2502 can be changed to include the corresponding number of line feed characters (CNTL J). For example, double spacing would be, PRINT@41:L$,"¬J".

Save Text (2680-2710)

In the save text routine, the user is given a chance to locate a file (lines 2680-2690), then the entire text is recorded (line 2700) as an ASCII data file, and the file is closed (line 2702). The program assumes the user has previously created a file of sufficient length to contain the text.

Delete line (2130-2194)

The delete line routine removes the characters comprising the current line from the text, T$ (see Figure 26). The current line is deleted one of two ways. If it's the last line of text (line 2182), the current line markers are redefined to be those of the previous line (line 2190), and the entire text is redefined to be all but the last line (line 2192). Then the routine ends (line 2194). This procedure is required since the replacement procedure (to be described) would generate invalid values for the current line markers (C,E). In other words, C would be greater than the number of characters in the entire text, T$. If the current line is not the last line, the line is removed by replacing with a null string (line 2184), and the ending marker is adjusted to account for the deleted characters (line 2186).

Insert Line (2030-2080)

The insert line routine allows the user to insert any number of lines before the current line (see Figure 27). The word, 'INSERT' is printed to indicate the routine has begun execution (line 2030). If a null string is entered, the routine ends (line 2045). Otherwise, a carriage return is concatenated on the line to be inserted (line 2046), the replacement is made without deleting any characters (line 2050), and the starting position of the current line is adjusted to account for the inserted line (line 2065). This process repeats until the user enters a null string.

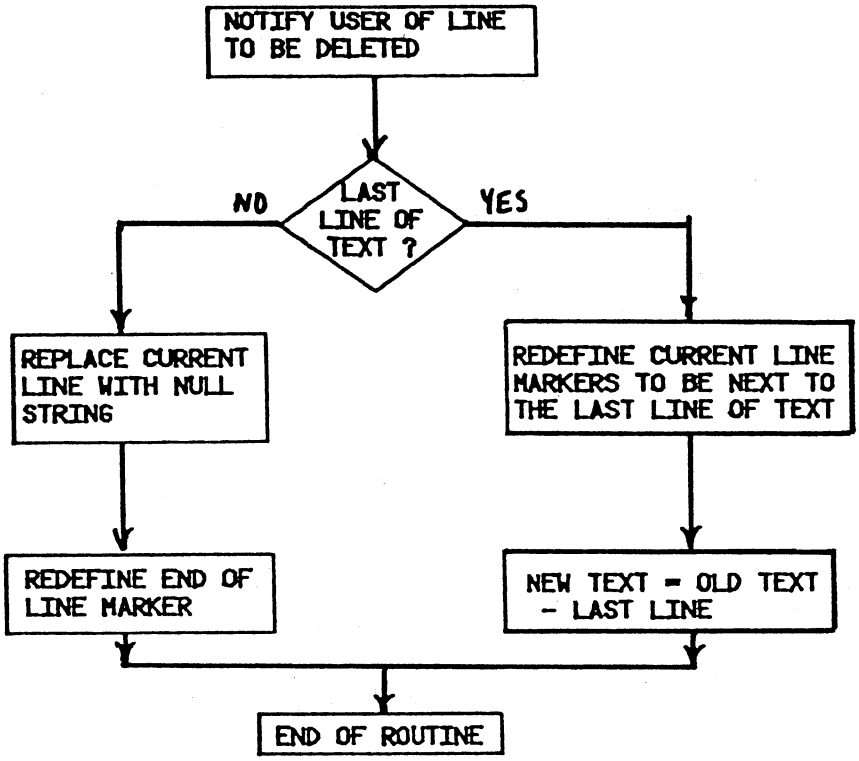


Figure 26. Flowchart of Delete Line Routine

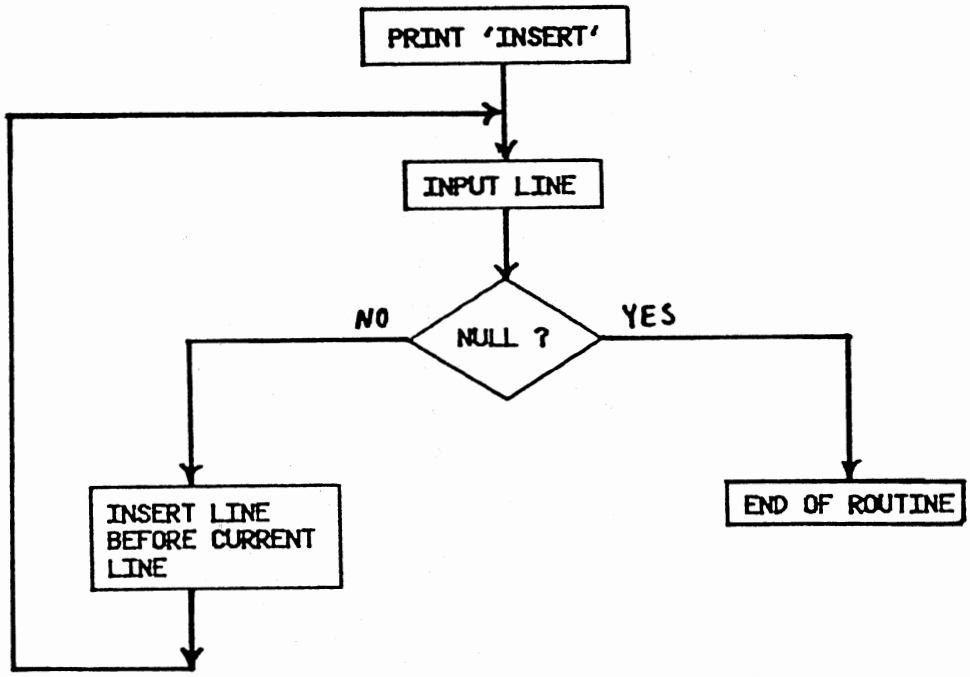


Figure 27. Flowchart of Insert Line Routine

Print Line Up (900-970)

The print line up routine uses a subroutine to define the current line markers (C and E) to be those of the previous line (line 960), and then jumps to the print line routine.

Print Line Down (1100-1165)

The print line down routine is exactly like the print line up routine except the move line down subroutine is used (line 1160).

Print Line (1500-1540)

The print line routine displays the current line. The current line, L$, is defined as the segment of the text, T$, begining at position C and ending at position E (line 1530). The semicolon, which inhibits the automatic carriage return, is used when displaying the line (line 1535) since the character at position E is a carriage return and part of L$.

Print Page Up (730-770)

The print page up routine displays 20 lines of text, starting with 20 lines previous to the current line. The move line up subroutine is called 20 times within a FOR-NEXT loop (lines 730-750) and the routine ends by jumping to the display page routine.

Print Page Down (810-850)

The print page down routine displays 20 lines of text begining with 20 lines after the current line. The routine is exactly like the print page up routine except the move down line subroutine is used (line 820).

Print Page 1 (890-892)

The print page 1 routine defines the current line markers, C and E, to be those of the first line of text (lines 890,891) and ends by jumping to the display page routine (line 892).

Find File Subroutine (10030-10080)

The find file subroutine allows a user to enter a file number, and then locates that file. The variable A$ is passed to the subroutine by either the save, print, receive, or send routines. A numeric quantity must be typed for the file number.

Move Up 1 Line (10250-10320)

This subroutine redefines the current line markers, C and E, to the values of the previous line . If the current line is the first line of text, the routine quits (line 10260). Otherwise, the new ending position is set to one ahead of the old begining postion (line 10262). Then the characters of the previous line, begining with the last character (line 10270), are searched for a carriage return

(line 10290). When a carriage return is detected, the new value of C is one more than the position of the carriage return.

Move Down 1 Line (10340-10370)

This routine redefines the current line markers to be those of the next line. The new begining of line marker, C, will be one more than the old end of line marker (line 10352). The new ending marker will be determined by the first carriage return character found searching forward from the new begining of line marker (line 10360).

Tektronix to IBM

Display Search Parameters (300-380)

This routine allows the user to view the current criterion the computer will use to recall the data. There are basically 2 types of criterion used in deciding which data to recall, they are the min-max range and the search parameter information. The first part displayed is the min-max range (lines 320-350). The string variable, A$, is a name of one of the variables which are defined in the data statement within the intialization (line 1032). A$ is passed to a subroutine which indentifies the variable for the operator, while displaying its recall status. If a variable is to be recalled, the min-max range is displayed. The string, S$, holds the search parameters which are compared to the parameter information associated with the lines of data recorded

on tape in order to decide if a particular line should be recalled.

Add Search Parameters (397-440)

The add search parameter routine allows the user to add parameter information which will be used in deciding which lines to recall. The parameter to be added to the list is input (lines 400,410). If the user enters a null string, the routine ends (line 412). Otherwise, the additional parameter (stored in A$) has a space concatenated for visual separation and then A$ is added to the end of S$, the search parameter string.

Delete Search Parameters (497-550)

The delete search parameter routine allows the user to remove parameters from the search parameter string, S$. The parameter to be deleted is input into A$ (lines 500,510). Then S$ is searched for the occurrence of A$ (line 520). If it can not be found, then the routine ends (line 530). Otherwise the parameter plus the space after it are replaced with a null string (line 540).

Search Routine (600-880)

The search routine is actually a subroutine (not activated by any user definable key) which is called by the search print, the search transfer, or the search display routines (see Figure 28). This subroutine is the main part

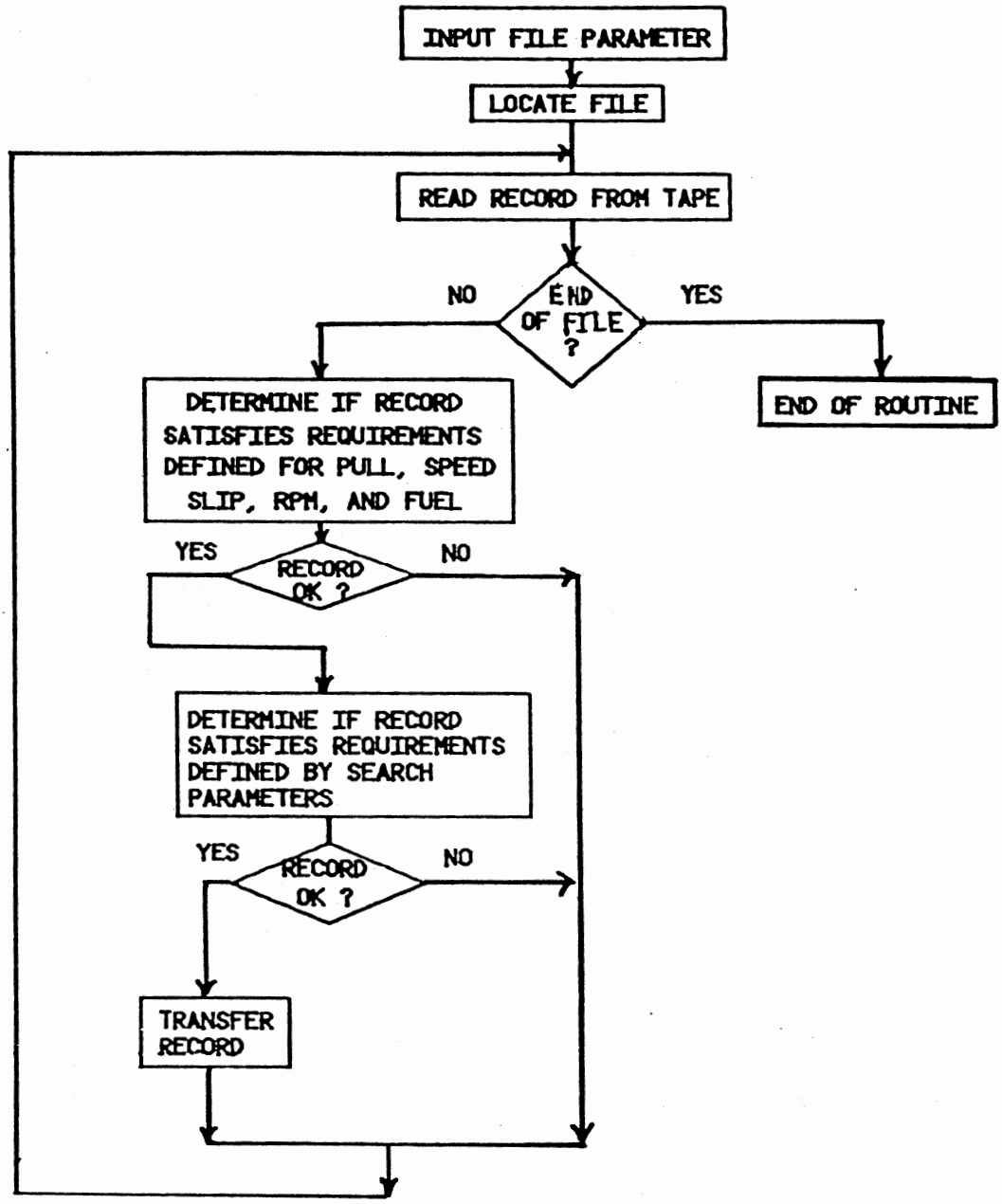


Figure 28. Flowchart of Search Routine

of the Tek to IBM program. There are basically two decisions made in recalling a line of data. The first one uses the min-max range and the second is based on the search parameter information. To begin with, the user is given the chance to indentify a file number (lines 600,610) and the computer locates the file (line 640). Since the search subroutine doesn't store the recalled data, it can operate on an infinitely sized file, ending only when the end of file marker is encountered (line 650).

The next section (lines 660,720) reads in a line of data from the tape and decides which variables (pull, speed, slip, rpm, fuel) should be recalled. Line 660 reads the line of data and stores it in the string variable, A$. The string L$ will hold the data to be recalled. Initially, it is nulled (line 662). The variable, I, is a counter which points to one of the 5 variables. Initially, it is set to 1 which points to pull (line 670). Then the string B$ is defined to be the characters representing the variable quantity pull (line 680). The characters are converted to a numeric quantity, called X (line 690). If pull has been deselected, then it isn't added to L$ and the program examines the next variable (line 692). If it is to be recalled, the lower and upper limits are checked to see if pull is within the allowable limit (line 700). If it isn't, the entire line is skipped, and the next line is investigated. If pull is within the limits, the characters are added to L$ (line 710), I is incremented to check the next variable

(line 712), and the above process repeats for all 5 variables (line 720). When the min-max requirements are met, the parameter information is examined.

The last part of the search subroutine checks the search parameter information (S$) against the parameter information recorded on tape and decides if the line should be recalled. The line, stored in A$, is redefined to be only the part which contains the parameter information (line 730). The variable B, initialized to 1, is used to indentify the starting position of each parameter within S$. The first parameter in S$ is isolated and stored in C$ (lines 750,760). Then the parameters which were read from tape, A$, are searched for the occurrence of the first parameter identified in the search parameter string, S$. If it can't be found, then the routine disregards the line in question (L$) and inputs the next line (line 780). If the line read from tape has a parameter which is listed in S$, then the process repeats for the next parameter of S$.

Assuming a line has been found which meets both variable requirements and the parameter requirements, the line is printed to 1 of 3 devices. If D1 is a 1, then the line is sent to the host computer connected to port 40 (line 850). The following input at port 40 statement waits for the line turn around characters indicating the host has accepted the line and is ready for another line. If the recalled line is not to be sent to the host computer, it is printed on either the printer (device 41) or the screen (device 32). The vari-

able D2 is either 41 or 32 accordingly.

Initialization (900-1040)

The arrays S and S1 are used in selecting the variables pull, speed, slip, fuel, rpm. Array S, dimensioned to 5, indicates if the variable should be recalled. A 1 indicates recall, a 0 indicates don't recall. The elements 1-5 correspond to the variables listed above, respectively. If a variable is to be recalled, the array S1 indicates the lower and upper limits which are acceptable. As with array S, the rows of S1 correspond to the variables. The first column of S1 holds the minimum value, the second column holds the maximum value. Initially, all the variables are selected (line 980) and the range is set from -99999 to 99999 (lines 990,1000). An internal parameter is set so the screen will automatically clear and home the cursor when the screen is full. Then the user function keys are enabled. Line 1020 sets an alternate line delimiter to a CNTL Q (decimal 19 on the ASCII chart) for use in the search transfer routine. The printer is initialized (line 1030) and a data statement is defined for use in the display search parameter routine.

Select/De-select Variable (100-250)

This subroutine allows the user to select which variables (pull,speed,fuel,slip,rpm) to recall and to define an acceptable range for these variables. The subroutine functions like a toggle switch. If the variable is not selected,

then the next time the variable key is pressed, the variable will be selected (line 140) and the user defines the min-max range for that variable (lines 150-180). If the variable is already selected, then it will become no longer selected (line 110).

Display Recall Status (200-250)

The display recall status subroutine prints on the screen a message indicating the variable is not selected or the min-max range if the variable is selected. The array, S1, indicates whether or not a variable is selected (line 210). If it is, the minimum and maximum values recorded in array S are displayed (line 240).

CHAPTER VI

PROGRAM OPERATION

Aim to Tektronix

Since the Aim 65 is used for other purposes than its role in the data management system, a fairly elaborate procedure is required to ensure proper operation in the Aim to Tek program. First, the Aim 65 program must be loaded into locations B000 to B048 from cassette tape. To do this, RAM must be mapped into B000 to BFFF by setting the memory bank switches on the mother board and positioning the jumpers on the underside of the Dram board to select B000-BFFF. The presence of RAM can be checked by writing and reading within the range B000-BFFF. If problems occur, check the continuity of the jumpers as they have been known to cause problems in the past. Once RAM has been located, the object code can be read from cassette using the 'L' command. The filename is 'TROBJ'. When the program is loaded, the code should match with the listing given in Appendix C. The next step is to set the baud rate to 2400 by storing 01 in location A417 and 5D in location A418. Then insert the data tape into the cassette player and rewind the tape. When the tape is rewound, press the '1' key which turns off the tape player via the remote line. Then depress the play button on the tape play-

er--the tape should not move. Finally, switch the TTY-KEYBOARD switch to the TTY position and press space bar.

Now the Aim to Tek program must be loaded from cartridge tape to 4052's memory. It is recommended the program be stored on the first file of a separate tape which always has the write protect feature enabled. If this precaution has been followed, the Aim to TEK cartridge tape can be inserted in the tape drive and the 'AUTOLOAD' key depressed to load and initialize the program. Once the Aim to Tek program is loaded, ensure the port 40 TBAR switch is positioned to 'Aim 65'. The Aim to Tek program is now ready for operation.

As an example of how to use the program, consider the following hypothetical case of a randomized block experiment conducted with a field cultivator at Blackwell, Oklahoma on June 3, 1981. The following text represents a typical sequence of events in using the Aim to Tek program to transfer data from cassette to cartridge tape. The user's action or typed response will be shown in lower case letters while the computer response will be in upper case letters.

User presses add file parameter key.

ENTER FILE PARAMETER-

User types: fc blackwell ok Jun 3 1881

CURRENT FILE PARAMETERS -

FC BLACKWELL OK JUN 3 1881

User presses delete file parameter key.

ENTER FILE PARAMETER TO DELETE -

User types: 1881

CURRENT FILE PARAMETERS-

FC BLACKWELL OK JUN 3

User presses add file parameter key.

ENTER FILE PARAMETER

User types: 1981

CURRENT FILE PARAMETERS

FC BLACKWELL OK JUN 3 1981

The operator is now ready to transfer data from cassette tape to cartridge tape.

User presses fetch key.

WAITING ON CASSETTE . . .

EXP UNIT TRANSFERRED.

GENERATING CALIBRATION CONSTANTS.

(The screen is cleared).

PULL	SPEED	FUEL	RPM	SLIP	
3450	4.1	6.3	2380	12.8	FC BLACKWELL OK JUN 3 1981
2710	4.3	6.2	2380	09.4	FC BLACKWELL OK JUN 3 1981
2890	4.2	6.2	2388	11.5	FC BLACKWELL OK JUN 3 1981
2690	4.0	6.3	2380	12.6	FC BLACKWELL OK JUN 3 1981
3220	4.3	6.5	2373	14.2	FC BLACKWELL OK JUN 3 1981
2580	4.2	6.3	2380	10.7	FC BLACKWELL OK JUN 3 1981
2320	4.3	6.2	2380	12.4	FC BLACKWELL OK JUN 3 1981
3090	4.4	5.9	2388	09.4	FC BLACKWELL OK JUN 3 1981
3110	4.3	6.0	2388	08.8	FC BLACKWELL OK JUN 3 1981
2870	4.3	6.0	2396	09.6	FC BLACKWELL OK JUN 3 1981

User presses add block parameter key.

ENTER BLOCK PARAMETER -

User types: 1 2

The 1 stands for the block number and the 2 stands for the replication number. The user could have entered these two numbers separately--using the add block parameter key

twice.

(The screen is cleared).

```
PULL SPEED FUEL RPM   SLIP
3450  4.1   6.3  2380  12.8 FC BLACKWELL OK JUN 3 1981 1 2
2710  4.3   6.2  2380  09.4 FC BLACKWELL OK JUN 3 1981 1 2
2890  4.2   6.2  2388  11.5 FC BLACKWELL OK JUN 3 1981 1 2
2690  4.0   6.3  2380  12.6 FC BLACKWELL OK JUN 3 1981 1 2
3220  4.3   6.5  2373  14.2 FC BLACKWELL OK JUN 3 1981 1 2
2580  4.2   6.3  2380  10.7 FC BLACKWELL OK JUN 3 1981 1 2
2320  4.3   6.2  2380  12.4 FC BLACKWELL OK JUN 3 1981 1 2
3090  4.4   5.9  2388  09.4 FC BLACKWELL OK JUN 3 1981 1 2
3110  4.3   6.0  2388  08.8 FC BLACKWELL OK JUN 3 1981 1 2
2870  4.3   6.0  2396  09.6 FC BLACKWELL OK JUN 3 1981 1 2
```

User presses store key.

ENTER FILE NUMBER TO STORE -

At this point the user should remove the Aim to Tek program tape and insert a data tape in the 4052's tape drive. The data tape should not be write protected and should have a new file of sufficient length to contain all of the experimental units which are to be recorded. In the worst case, each experimental unit would require about 750 bytes. Suppose file 1 of the data tape is to be used.

User types: 1

(Tape moves locating file 1)

EXP UNIT 1 HAS BEEN RECORDED.

User presses fetch key a second time.

EXP UNIT TRANSFERRED.

GENERATING CALIBRATION CONSTANTS.

Then the experimental unit is displayed as before, and the user adds a block and replication number as before only a different set of numbers, perhaps 3 and 4, for example.

User presses store key a second time.

IS THIS THE LAST EXP UNIT TO BE RECORDED ?

User types: no

EXP UNIT 2 HAS BEEN RECORDED.

The above process repeats for all experimental units to be transferred. Although the user never pressed the display key, the fetch and add block parameter routines end by jumping to the display routine. When the last experimental unit is recorded, the file is closed and the program re-initializes.

Editor

The Editor program has the most functions, but is not the most complicated of the 3 programs in the data management system. Like the Aim to Tek program, the Editor program should be stored on file 1 of a separate tape which always has the write protect feature enabled. If this is done, the Editor tape can be placed in the 4052's tape drive and the autoload key pressed. When the 4052 is no longer busy, the Editor tape can be removed and the data tape to be edited put in its place. The following sequence of events depict a typical example of the use of the Editor program. As in the Aim to Tek operation, all user responses or action will be typed in lower case, while computer responses will be in uppercase.

User presses recall key.

ENTER FILE TO RECALL -

User types: 1

(tape drive moves to file 1, then the data is read into memory)

User presses page 1 key.

(screen is cleared)

```
PULL SPEED FUEL RPM   SLIP
3450  4.1  6.3  2380  12.8 FC BLACKWELL OK JUN 3 1981 1 2
2710  4.3  6.2  2380  09.4 FC BLACKWELL OK JUN 3 1981 1 2
2890  4.2  6.2  2388  11.5 FC BLACKWELL OK JUN 3 1981 1 2
2690  4.0  6.3  2380  12.6 FC BLACKWELL OK JUN 3 1981 1 2
3220  4.3  6.5  2373  14.2 FC BLACKWELL OK JUN 3 1981 1 2
2580  4.2  6.3  2380  10.7 FC BLACKWELL OK JUN 3 1981 1 2
2320  4.3  6.2  2380  12.4 FC BLACKWELL OK JUN 3 1981 1 2
3090  4.4  5.9  2388  09.4 FC BLACKWELL OK JUN 3 1981 1 2
3110  4.3  6.0  2388  08.8 FC BLACKWELL OK JUN 3 1981 1 2
2870  4.3  6.0  2396  09.6 FC BLACKWELL OK JUN 3 1981 1 2
3450  4.1  6.3  2380  12.8 FC BLACKWELL OK JUN 3 1981 3 4
2710  4.3  6.2  2380  09.4 FC BLACKWELL OK JUN 3 1981 3 4
2890  4.2  6.2  2388  11.5 FC BLACKWELL OK JUN 3 1981 3 4
2690  4.0  6.3  2380  12.6 FC BLACKWELL OK JUN 3 1981 3 4
3220  4.3  6.5  2373  14.2 FC BLACKWELL OK JUN 3 1981 3 4
2580  4.2  6.3  2380  10.7 FC BLACKWELL OK JUN 3 1981 3 4
2320  4.3  6.2  2380  12.4 FC BLACKWELL OK JUN 3 1981 3 4
3090  4.4  6.1  2388  09.4 FC BLACKWELL OK JUN 3 1981 3 4
3110  4.3  6.0  2388  08.8 FC BLACKWELL OK JUN 3 1981 3 4
2870  4.3  6.0  2396  09.6 FC BLACKWELL OK JUN 3 1981 3 4
```

User presses line key.

```
3450  4.1  6.3  2380  12.8 FC BLACKWELL OK JUN 3 1981 1 2
```

User presses line down key.

```
2710  4.3  6.2  2380  09.4 FC BLACKWELL OK JUN 3 1981 1 2
```

User presses line up key.

```
3450  4.1  6.3  2380  12.8 FC BLACKWELL OK JUN 3 1981 1 2
```

Suppose the user has decided that the file parameter, 1981, is not really necessary. In this case, the find/replace can be used to remove it.

User presses find/replace key.

ENTER (STRING TO FIND/STRING TO REPLACE/NUMBER OF REPEATS)-

User types: 1982//10

There would be no response from the computer because the user has mistakenly typed in 1982 for 1981. The computer can not find the string 1982.

User presses find/replace key.

ENTER (STRING TO FIND/STRING TO REPLACE/NUMBER OF REPEATS)-

User types: 1981//10

```
PULL SPEED FUEL RPM   SLIP
3450  4.1   6.3  2380  12.8 FC BLACKWELL OK JUN 3 1 2
2710  4.3   6.2  2380  09.4 FC BLACKWELL OK JUN 3 1 2
2890  4.2   6.2  2388  11.5 FC BLACKWELL OK JUN 3 1 2
2690  4.0   6.3  2380  12.6 FC BLACKWELL OK JUN 3 1 2
3220  4.3   6.5  2373  14.2 FC BLACKWELL OK JUN 3 1 2
2580  4.2   6.3  2380  10.7 FC BLACKWELL OK JUN 3 1 2
2320  4.3   6.2  2380  12.4 FC BLACKWELL OK JUN 3 1 2
3090  4.4   5.9  2388  09.4 FC BLACKWELL OK JUN 3 1 2
3110  4.3   6.0  2388  08.8 FC BLACKWELL OK JUN 3 1 2
2870  4.3   6.0  2396  09.6 FC BLACKWELL OK JUN 3 1 2
```

Since the number of repeats was specified to 10, only 10 lines have the 1981 removed. The user could have removed the 1981 from the entire data set if the number of repeats was equal to the number of lines in the data set. Suppose this is done, and the user now wants to transfer the entire data set to the host computer. First, the data in memory must be saved on tape.

User presses store key.

ENTER FILE TO SAVE -

User types: 1

(Then the computer locates file 1 and records the data).

User presses send key.

ENTER FILE TO SEND -

Before entering the file to send, the TBAR switch on port 40 should be switched to the plotter position and the

plotter should be turned off.

User types: 1

(Computer locates file 1)

LOGON...ENTER EDIT MODE.... PRESS RETURN TO BASIC

The busy and the I/O lights on the 4052 should now be alternately flashing off and on. The 4052 is waiting for connection to the host computer. To do this, turn on the modem, dial 7600 (1200 baud campus line), and place the phone handset in the modem. When the connection is established, the busy and the I/O lights should stop flashing and the carrier detect light (CXD) on the modem should turn on. When this occurs, press the space bar and then a carriage return. The host computer (IBM 370) should respond with the message ENTER LOGON. Now logon in the usual manner, enter the edit session on a new file, i.e., the subcommand INPUT, and press the return to basic key. The data stored on 4052's tape file 1 should now be transferring to the host computer. When the busy light on the 4052 goes off, the transfer is complete. During times of heavy computer use, the IBM may put the TSO session on hold and the transfer seemingly stop. If this happens and the busy light has not gone off, the 4052 is waiting for a signal from the host indicating readiness to receive another line of data. In order to save the data on the host computer, the call termin button must be pushed, and then a carriage return which places the TSO session in the edit mode, and then the word 'SAVE' must be entered, followed by a carriage return. To return back to

the Editor program on the 4052, press the return to basic key again.

There are 2 ways the receive key can be used. The receive key can be used to transfer a program or dataset from the host to the 4052 tape and to transfer the output from a program on the IBM to 4052's tape. Like the send key, pressing the receive key causes the 4052 to ask for the file number to receive. This file should be a new ASCII file or at least contain ASCII data which can be destroyed. The file should also be of sufficient length to contain the data to be sent from the IBM. When the file number is entered, the 4052 will type, LOGON....ENTER COMMAND LINE.....RETURN TO BASIC. If connection to the host has already been established, there is no need to repeat the logon procedure described earlier. If this is the case, the 4052 will be emulating an ASCII terminal connected to the IBM 370. If a data set of 72 characters or less per line is to be received then type 'LIST DATASET.NAME' but don't hit carriage return. When the return to basic key is pressed, the carriage return will be sent to the IBM and the 4052 will begin to accept the ASCII characters and record them on tape. When the character sequence 'READY' is encountered, the receive routine ends.

If program output of 132 characters per line is to be received, then type 'TERMINAL LINESIZE(132)' while the 4052 is acting as an ASCII terminal. The IBM should respond with 'READY'. Now type the command line which would normally

cause the output to be displayed on the screen. For example, if a SAS job is to be executed interactively, this could be '%SAS DS(DATASET.CNTL)'. If a job has already executed and the output is held in a output queue, then type 'OUTPUT JOBNAME(JOBID)'. In both cases, don't hit carriage return after typing in the command line which causes the output to be displayed. Instead, push the return to basic key which causes a carriage return to be sent and then the 4052 immediately goes into the receive mode . After the receive routine ends, the print file command can be used to generate a printed copy of the output.

Tektronix to IBM

Like the other two programs, the TEK to IBM program should be on a separate tape at file 1. If this is the case, the tape can be inserted and the autoload key pressed. When the program is in memory, remove the tape and insert the data tape to be searched. Suppose data for pull and fuel flow for all tests conducted in June are to be examined. A typical sequence of events to generate this subset of data might be as follows:

User presses speed key.

SPEED HAS BEEN DE-SELECTED

User presses RPM key.

RPM HAS BEEN BEEN DE-SELECTED

User presses slip key.

SLIP HAS BEEN DE-SELECTED.

User presses add search parameter key.

ENTER SEARCH PARAMETER -

User types: jun

CURRENT SEARCH PARAMETERS -

JUN

User presses display key.

(screen is cleared)

PULL MINIMUM=-9999 PULL MAXIMUM=9999

FUEL MINIMUM=-9999 FUEL MAXIMUM=9999

RPM HAS BEEN DESELECTED

SPEED HAS BEEN DESELECTED

SLIP HAS BEEN DESELECTED

CURRENT SEARCH PARAMETERS-

JUN

Now suppose the user wants a printed copy of this data set (pull and fuel values for the month of June). To do this, simply press the search print.

User presses search print key.

ENTER FILE NUMBER TO SEARCH.

User types: 1

INSERT TAPE, HIT RETURN

User hits return (tape has already been inserted).

Then the computer performs the sequential search described in an earlier chapter. When a line of data on tape is encountered which meets the requirements, it is printed. If the user has other files on which the same search is to be performed, then simply press the search print key again,

only this time entering the appropriate tape and file number. If the selected data is to be transferred to the host computer, establish connection as described earlier in this chapter and use the search transfer key instead of the search print key. Now, suppose the user wants to generate a data set like the one just generated only for values of pull within the range 1500 to 2000.

User presses pull key.

PULL HAS BEEN DESELECTED.

User presses pull key.

ENTER MINIMUM VALUE OF PULL-

User types: 1500

ENTER MAXIMUM VALUE OF PULL-

User types: 2000

The program is now ready to perform either the search transfer, search print, or the search display.

CHAPTER VII

SUMMARY AND DISCUSSION

This thesis describes a data management system designed to store tractor performance data. A microcomputer based tractor performance data acquisition instrument records data taken in the field onto cassette tape. The cassette tape is then taken into the laboratory where the data management system resides. Data is transferred from cassette tape to cartridge tape, where it can be accessed by the data management system. As data is transferred, the experimenter can add descriptive information. This descriptive information forms a basis for selecting particular parts of the entire database for analysis. Analysis can be aided by transferring all or part of the data to the University Computer, where statistical and graphical software languages can be utilized.

One major limitation of the system is that the magnetic tape cartriges where the data is stored can only hold 3 tenths of a megabyte. Experience has shown this is more than enough space to contain all the data generated in a year. This may not always be the case as modifications to the acquisition system may generate more numbers, or the existing acquisition system may be used more often.

Despite the disadvantage of storage limitations imposed

by cartridge tapes, there are also advantages of building the database on the cartrige tapes, i.e., keeping the database within the Agricultural Engineering Department. One of these advantages is being free from the University Computer's operating hours (scheduled or unscheduled) and the possibility of inadvertent destruction of the data. Currently, usage of the University Computer is popular enough to cause waits of up to several minutes for response. If the system goes down, any database on the system is totally inaccessible. Some catastrophic computer failures could destroy the contents of a data tape, although much of this risk is relieved by making backup copies of the tape weekly, for storage in vaults within the department.

Another advantage to building the database on cartridge tapes is the timelessness of being able to use the University Computer or other university computers. Researchers who had databases on tapes which operated on the IBM 360 (previous University Computer) were unable to use them when the University converted to an IBM 370. Since the University Computer should always have a remote link no matter what changes are made, the database will always be able to be transferred to the University Computer. Moreover, the database could be transferred to any computer capable of supporting a telephone link. This represents an exciting future for Agricultural Engineering--the distributed database. Eventually, universities throughout the United States will have access to data acquired anywhere else in the Country,

allowing a unified research effort.

Finally, having the database within the Department allows for the opportunity of in house software development. Since the Tektronix 4052 is a BASIC language graphics microcomputer, programs could be written which analyze and graph the data, thus eliminating the dependency on the University Computer.

Thus far, the system has only been used as a way to get the data into the University Computer. In other words, none of the data is being incorporated into the database. This is probably due to the fact the data is needed only for immediate analysis, and need not be permanently stored. It may be due to the operator's preference of another way to manage the data or the operator being unfamiliar with the system.

CHAPTER VIII

CONCLUSIONS

The data management system performs all of the functions listed in Chapter II. The system is tailored to accept data recorded on cassette tape by a microcomputer based data acquisition system. By invoking the fetch routine, data is transferred from cassette tape into the system memory. Another routine allows descriptive information to be added to data in system memory. A store function allows the data, along with the additional descriptive information, to be incorporated into the database. A general purpose editor allows the operator to alter the database. The Editor program also allows interactive transfer of data files to or from a host computer. Another program allows the user to select particular parts of the data to be transferred.

CHAPTER IX

RECOMMENDATIONS

The most limiting aspect of the Data Management System is the size of the cartridge tapes. Each tape can hold approximately 300 Kbytes. So, if the database gets as large as 6 Mbytes, it could mean interchanging 20 cartridge tapes when performing searches. The solution to this problem is the addition of a hard disk drive. Modification to the system would involve changing the present routines that access the cartridge tapes. Files are presently identified by number. Depending on the file manager accompanying the hard drive, the file numbers may need to be changed to file names. The routines effected are the store routine of the Aim to Tektronix program, the recall, store, send, receive, and print routines of the Editor program, and the search routines of the Tektronix to IBM program.

Another limitation is the 1200 baud modem. If data sets as large as a megabyte are to be transmitted, it will take over two hours to transfer. A faster modem may alleviate some of this problem but the telephone lines may not support increased transfer rates. Establishing a direct link may be a possibility. Another solution would be to set the transfer to begin automatically in the middle of the night. This would involve automatic dialing and predetermined dia-

log between the Tektronix and the IBM Computer.

Another attractive change would be to completely describe the experimental unit in the field. Thus, when the data is transferred into the system, it could be automatically recorded into the database. This change would require altering the Aim to Tektronix program. In addition, a slight modification of the Aim to Tektronix program could allow the Management System to be used by other data acquisition systems.

By using the routines to selectively recall data, graphical and statistical functions could be incorporated, thus making the system more independent. Although there are exsiting statistical and graphical routines available for the Tektronix, it would mean extensive programming to get the data into the correct format so these software packages could be used. Instead, it would be simpler to develop a line or bar chart plotting routine and a statistical routine to calculate sum of squares. There may be some plotting or statistical software which would require little modification of the data, but such software is unknown at this time.

REFERENCES

1. Hohenberger, J. G. and P.G. Alexander. 1981. Microprocessor Based Data Acquisition System For Mobile Equipment. Paper No. 81-1569. ASAE, St. Joseph, MI 49085.

2. Garner, T.H. 1980. Tillage Mechanical Energy Input and Soil-Crop Response. Paper No. 80-1026. ASAE, St. Joseph, MI 49085.

3. Hendrik, J.G., C.E. Johnson, R.L. Schafer, J.D. Jarrell. 1981 A Microprocessor-Based Field Data Acquisition System. Paper No. 81-1577. ASAE, St. Joseph, MI 49085.

4. Tompkins, F.D. and L.R. Wilhelm. 1981. Instrumentation for Measuring Energy Inputs to Implements. Paper No. 81-1575, ASAE, St. Joseph, MI 49085.

5. Grevis-James, I.W. , D.R. DeVoe, P.D. Bloome, D.G. Batchelder 1981. Microcomputer Based Data Acquisition System for Tractors. Paper No. 81-1578. ASAE. St. Joseph, MI 49085.

6. Draper, Jesse M. 1981. Cost and Benefits of Database Management: Federal Experience. NBS Special Publication 500-84. National Bureau of Standards, U.S. Department of Commerce.

7. Zemrowski, Ken. 1976. Distributed Data Base Concepts in The Technology of Data Base Management Systems by Richard A. Bassler and Jimmie J. Logan. College Readings Inc. Alexandria, Virginia

8. Jacobs, Barry E. April 2, 1982. "On Database Logic" in Journal of the Association for Computing Machinery, Vol. 29, No.2, pp. 310-332.

9. DBMASTER. 1982. Stoneware Software Company, 1930 Fourth St., San Rafael, CA., 94901

10. PROFILE PLUS. 1982. Tandy Corporation, Fort Worth, Texas, 76102

11. DATA COMMUNICATION INTERFACE Operator's Manual. 1976. Part No. 070-2066-01. Tektronix, Inc. Beaverton, Oregon. 1976

APPENDIXES

APPENDIX A

AIM TO TEKTRONIX PROGRAM

```
1 REM ****************************JUMP PAST SUBROUTINES
2 GO TO 1000
3 REM*****************************************************
4 REM ********************************** DISPLAY BLOCK
5 GO TO 3010
8 REM ************************************ FETCH BLOCK
9 GO TO 2010
12 REM ********************************** ADD COLUMN
13 GO TO 5010
16 REM********************************** STORE BLOCK
17 GO TO 4030
32 REM****************************** ADD FILE PARAMETER
33 GO TO 8030
52 REM********************************** DELETE COLUMN
53 GO TO 6010
72 REM ************************* DELETE FILE PARAMETER
73 GO TO 7030
100 REM*****************************************************
110 REM ****************************** BEGIN SUBROUTINES
120 REM*****************************************************
130 REM
140 REM*************** hex string (h$) into decimal (d)
150 D=0
160 X2=0
170 X1=LEN(H$)
180 G$=SEG(H$,X1,1)
190 X=ASC(G$)
200 IF X<48 OR (X>57 AND X<65) OR X>70 THEN 270
210 X=X-55+(X<65)*7
220 D=D+X*16**X2
230 X2=X2+1
240 X1=X1-1
250 IF X1>0 THEN 180
260 RETURN
270 PRINT "INVALID HEXADECIMAL CHARACTER "
280 RETURN
290 REM
300 REM
310 REM****************************** ON SIZE INTERRUPT
320 X=9999
340 RETURN
350 REM********************* subroutine to activate aim
360 PRINT @40:"5"
370 CALL "wait",0.3
380 PRINT @40,30:
390 S9=1
400 REM s9=0 says first block has been fetched.
410 RETURN
1000 REM ******************************************
1010 REM ************************PROGRAM INITIALIZATIO
1020 REM ******************************************
1030 Z$=" PULL  FUEL  RPM   GND   SLIP"
1040 PRINT @40,30:
1050 CALL "RATE",2400,2,2
```

```
1060 E9=1
1070 T9=0
1080 S9=0
1090 M$=CHR(13)
1100 SET CASE
1110 PAGE
1120 DELETE T,P,F,R,G,S,W,C,B$,D$,E$
1130 DIM T(60),P(10),F(10),R(10),G(10)
1140 DIM S(10),W(10),C(10),B$(255)
1150 DIM D$(720),E$(720)
1160 D$=""
1170 E$=""
1180 P$=""
1190 SET KEY
1200 END
2000 REM************************************************
2010 REM********************************** FETCH BLOCK
2020 REM************************************************
2030 D$=""
2040 B$=""
2050 IF S9=1 THEN 2110
2060 PRINT "Waiting on cassette . . ."
2070 REM ++++++++++++++++++++++++++++++++++++++++++++
2080 REM  BEGIN EXP. UNIT TRANSFER
2090 PRINT @40,30:
2100 GOSUB 350
2110 INPUT @40,0:A
2120 IF A=0 THEN 2110
2130 REM++++++++++++
2140 INPUT @40:B$
2150 PRINT @40,30:
2160 IF LEN(B$)=234 THEN 2230
2170 REM++++++++++++
2180 PRINT "ERROR. EXP UNIT NOT TRANSFERRED."
2190 B$=""
2200 GOSUB 350
2210 RETURN
2220 REM++++++++++++
2230 PRINT "EXP UNIT TRANSFERRED"
2240 GOSUB 350
2250 PRINT "Calculating . . . "
2255 REM + HEX STRING B$ TO DECIMAL T
2260 J=1
2270 FOR I=1 TO 197 STEP 4
2280 H$=SEG(B$,I+2,2)
2290 A$=SEG(B$,I,2)
2300 H$=H$&A$
2310 GOSUB 150
2320 T(J)=D
2330 J=J+1
2340 NEXT I
2350 FOR I=201 TO 225 STEP 4
2360 H$=SEG(B$,I,4)
2370 GOSUB 150
```

```
2380 T(J)=D
2390 J=J+1
2400 NEXT I
2410 H$=SEG(B$,229,6)
2420 GOSUB 150
2430 T(J)=D
2440 REM++++++++++++
2540 REM++++++++++++
2550 REM
2560 REM  APPLY CALIBRATION CONSTANTS - GENERATE SLIP
2570 ON SIZE THEN 320
2580 FOR I=1 TO 10
2590 X=(T(I)-T(52))*20000/(T(51)-T(52))
2600 T(I)=INT(X)
2610 X=T(58)/T(I+20)
2620 T(I+20)=INT(X)
2630 X=T(57)/T(I+10)
2640 T(I+10)=INT(100*X)/100
2650 X=(1-T(I+30)/T(I+40)*T(55)/T(56))*100
2660 T(I+40)=INT(X*10)/10
2670 X=T(I+30)*262.24/T(54)
2680 T(I+30)=INT(X*100)/100
2690 NEXT I
2810 FOR J=1 TO 10
2830 A$=""
2840 FOR I=1 TO 5
2850 G$=STR(T(J+(I-1)*10))
2860 IF LEN(G$)=>6 THEN 2890
2870 G$=G$&" "
2880 GO TO 2860
2890 A$=A$&G$
2900 NEXT I
2910 D$=D$&A$
2920 D$=D$&M$
2930 NEXT J
2940 REM++++++++++++
2950 A$=P$
2960 GO TO 5070
3000 REM ************************************************
3010 REM ********************************* DISPLAY BLOCK
3020 REM************************************************
3030 REM
3040 PAGE
3050 PRINT Z$
3060 PRINT D$
3070 RETURN
4000 REM************************************************
4010 REM ********************************** Store block
4020 REM************************************************
4030 IF T9<>0 THEN 4080
4040 PRINT "ENTER FILE NUMBER TO RECORD DATA -";
4050 INPUT F1
4060 FIND F1
4070 GO TO 4130
```

```
4080 PRINT "IS THIS THE LAST EXP UNIT TO BE RECORDED";
4090 INPUT A$
4100 IF A$<>"Y" THEN 4130
4110 PRINT @33:D$;
4120 GO TO 1030
4130 PRINT @33:D$;
4140 PRINT "EXP UNIT ";E9;" HAS BEEN RECORDED."
4150 E9=E9+1
4160 T9=1
4170 RETURN
5000 REM************************************************
5010 REM ********************** add columns to exp unit
5020 REM************************************************
5030 PRINT "ENTER COLUMN VALUE -";
5040 INPUT A$
5050 IF A$="" THEN 3010
5060 A$=A$&" "
5070 X=1
5080 X1=POS(D$,M$,X)
5090 IF X1-X+LEN(A$)=>71 THEN 5140
5100 D$=REP(A$,X1,0)
5110 X=X1+LEN(A$)+1
5120 IF X<LEN(D$) THEN 5080
5130 GO TO 3010
5140 PRINT "CANT ADD COLUMN VALUE--LINE TOO LONG."
5150 RETURN
6000 REM************************************************
6010 REM ************************** delete column value
6020 REM************************************************
6030 PRINT "Enter column value to delete -";
6040 INPUT A$
6050 X=POS(D$,M$,1)
6060 X1=POS(D$,A$,31)
6070 IF X1=0 OR X1>X THEN 3010
6080 FOR I=1 TO 10
6090 D$=REP("",X1,LEN(A$))
6100 X1=X1+X-LEN(A$)
6110 NEXT I
6120 GO TO 3010
7000 REM************************************************
7010 REM************************* DELETE FILE PARAMETER
7020 REM************************************************
7030 PRINT "ENTER FILE PARAMETER TO DELETE - ";
7040 INPUT A$
7050 X=POS(P$,A$,1)
7060 IF X=0 THEN 8080
7070 P$=REP("",X,LEN(A$)+1)
7080 GO TO 8080
8000 REM************************************************
8010 REM***************************** ADD FILE PARAMETER
8020 REM************************************************
8030 PRINT "ENTER FILE PARAMETER - ";
8040 INPUT A$
8045 IF A$="" THEN 8080
```

```
8050 A$=A$&" "
8060 IF LEN(A$)+LEN(P$)=>30 THEN 8100
8070 P$=P$&A$
8080 PRINT "CURRENT FILE PARAMETERS:";P$
8090 RETURN
8100 PRINT "CANT ADD FILE P--NO ROOM FOR BLOCK PS"
8110 RETURN
```

APPENDIX B

EDITOR PROGRAM

```
1 GO TO 4440
2 REM*******************************************
3 REM         --------- USER DEFINED KEYS -------
4 SET NOKEY
5 GOSUB LEN(T$)>0 OF 730
6 SET KEY
7 RETURN
8 SET NOKEY
9 GOSUB LEN(T$)>0 OF 810
10 SET KEY
11 RETURN
12 SET NOKEY
13 GOSUB LEN(T$)>0 OF 890
14 SET KEY
15 RETURN
16 SET NOKEY
17 GOSUB LEN(T$)>0 OF 100
18 SET KEY
19 RETURN
24 SET NOKEY
25 GOSUB LEN(T$)>0 OF 960
26 SET KEY
27 RETURN
28 SET NOKEY
29 GOSUB LEN(T$)>0 OF 1160
30 SET KEY
31 RETURN
32 SET NOKEY
33 GOSUB LEN(T$)>0 OF 510
34 SET KEY
35 RETURN
36 SET NOKEY
37 GOSUB LEN(T$)>0 OF 1530
38 SET KEY
39 RETURN
40 SET NOKEY
41 GOSUB 2030
42 SET KEY
43 RETURN
44 SET NOKEY
45 GOSUB 2680
46 SET KEY
47 RETURN
48 SET NOKEY
49 GOSUB 4000
50 SET KEY
51 RETURN
52 SET NOKEY
53 GOSUB 2462
54 SET KEY
55 RETURN
56 PRINT @40:"off";M$
57 PAGE
58 CALL "termin"
```

```
59 RETURN
60 CALL "TERMIN"
62 SET KEY
63 RETURN
64 SET NOKEY
65 GOSUB 3000
66 SET KEY
67 RETURN
68 SET NOKEY
69 GOSUB 3740
70 SET KEY
71 RETURN
72 SET NOKEY
73 GOSUB LEN(T$)>0 OF 2120
74 SET KEY
75 RETURN
76 SET NOKEY
77 GOSUB 6000
78 SET KEY
79 RETURN
80 SET NOKEY
81 GOSUB 5000
82 SET KEY
83 RETURN
90 REM*****************************************
91 REM    -------------- PRINT PAGE -------------
92 REM *****************************************
100 PAGE
110 I=1
112 C1=C
120 IF E=LEN(T$) OR I=20 THEN 180
130 L$=SEG(T$,C,E-C+1)
140 PRINT L$;
150 GOSUB 10350
160 I=I+1
170 GO TO 120
180 L$=SEG(T$,C,E-C+1)
190 PRINT L$;
200 C=C1
210 E=POS(T$,M$,C)
230 RETURN
480 REM*********************************************
490 REM                        FIND/REPLACE
500 REM*********************************************
510 PRINT "ENTER (find/replace/number of repeats)"
512 INPUT A$
514 X1=POS(A$,"/",1)
516 X2=POS(A$,"/",X1+1)
518 IF X1=0 OR X1=1 OR X2=0 OR X2=2 THEN 660
520 F$=SEG(A$,1,X1-1)
522 R$=SEG(A$,X1+1,X2-X1-1)
524 IF X2=LEN(A$) THEN 532
526 N$=SEG(A$,X2+1,LEN(A$)-X2)
528 R=VAL(N$)
```

```
530 IF R<>0 THEN 590
532 R=1
590 I=1
600 F=POS(T$,F$,C)
610 IF F=0 THEN 660
615 IF F>E THEN 642
619 I=I+1
620 T$=REP(R$,F,LEN(F$))
621 E=E+LEN(R$)-LEN(F$)
622 L$=SEG(T$,C,E-C+1)
624 PRINT L$;
625 C=F+LEN(R$)
630 IF I<=R THEN 600
642 GOSUB 10350
650 IF I<=R THEN 600
655 GOSUB 10260
660 RETURN
700 REM*********************************************
710 REM       PRINT PAGE UP
720 REM*********************************************
730 FOR I=1 TO 20
740 GOSUB 10260
750 NEXT I
770 GO TO 100
780 REM*********************************************
790 REM                  PRINT PAGE DOWN
800 REM*********************************************
810 FOR I=1 TO 20
820 GOSUB 10350
830 NEXT I
850 GO TO 100
860 REM*********************************************
870 REM         PRINT PAGE 1
880 REM*********************************************
890 C=1
891 E=POS(T$,M$,C)
892 GO TO 100
900 REM***************************************
920 REM---------------------- PRINT LINE UP
940 REM***************************************
960 GOSUB 10260
970 GO TO 1530
1100 REM***************************************
1120 REM---------------------- PRINT LINE DOWN
1140 REM***************************************
1160 GOSUB 10350
1165 GO TO 1530
1500 REM***************************************
1510 REM                  PRINT LINE
1520 REM***************************************
1530 L$=SEG(T$,C,E-C+1)
1535 PRINT L$;
1540 RETURN
2000 REM***************************************
```

```
2010 REM                              INSERT A LINE
2020 REM*****************************************
2030 PRINT "INSERT -"
2040 INPUT A$
2045 IF A$="" THEN 2080
2046 A$=A$&M$
2050 T$=REP(A$,C,0)
2065 C=C+LEN(A$)
2070 GO TO 2040
2080 GOSUB 10260
2082 RETURN
2090 REM*****************************************
2100 REM                            DELETE A LINE
2110 REM*****************************************
2120 REM
2130 PRINT "LINE DELETED IS . . ."
2136 L$=SEG(T$,C,E-C+1)
2140 PRINT L$;
2182 IF E=LEN(T$) THEN 2190
2184 T$=REP("",C,LEN(L$))
2186 E=POS(T$,M$,C)
2188 RETURN
2190 GOSUB 10260
2192 T$=SEG(T$,1,E)
2200 RETURN
2420 REM*****************************************
2440 REM---------------------- PRINT FILE
2460 REM*****************************************
2462 A$="PRINT"
2472 GOSUB 10050
2473 PRINT @41,11:1
2482 ON EOF (0) THEN 2506
2492 INPUT @33:L$
2502 PRINT @41:L$
2503 GO TO 2492
2506 END
2620 REM*****************************************
2640 REM---------------------- SAVE TEXT TO FILE
2660 REM*****************************************
2680 A$="SAVE"
2690 GOSUB 10050
2700 PRINT @33:T$;
2702 CLOSE
2710 RETURN
2940 REM*****************************************
2960 REM---------------------- RECALL TEXT FROM FILE
2980 REM*****************************************
3000 A$="RECALL"
3010 GOSUB 10050
3020 ON EOF (0) THEN 3060
3030 INPUT @33:L$
3040 T$=T$&L$
3042 T$=T$&M$
3050 GO TO 3030
```

```
3060 C=1
3070 E=POS(T$,M$,C)
3072 END
3680 REM*****************************************
3700 REM------------------- RECEIVE GENERAL
3720 REM*****************************************
3740 A$="RECIEVE"
3742 GOSUB 10050
3760 CALL "MARGIN",0,0,1
3780 CALL "TCRLF",1,0,0
3800 CALL "EOLCHR",13,M$,0
3820 CALL "RATE",1200,2,2
3840 CALL "RSTRIN","",M$,"READY"
3860 PRINT "LOGON-ENTER COMMAND LINE-RET TO BASIC."
3880 CALL "TERMIN"
3900 CALL "DTRECV"
3920 RETURN
3940 REM*****************************************
3960 REM-------------------- SEND
3980 REM*****************************************
4000 A$="SEND"
4010 GOSUB 10050
4180 CALL "PROMPT",1,500," "
4200 CALL "TCRLF",1,0,0
4220 CALL "TSTRIN","","",""
4240 CALL "RSTRIN","","",""
4260 CALL "RATE",1200,2,2
4280 CALL "EOLCHR",13,"",0
4300 PRINT "LOGON-ENTER EDIT MODE-RETURN TO BASIC."
4320 CALL "TERMIN"
4340 CALL "DTSEND"
4360 RETURN
4380 REM*****************************************
4400 REM---------------------- INITIALIZE
4420 REM*****************************************
4440 DELETE T$,L$
4450 DIM T$(45000),L$(132)
4460 T$=""
4465 M$=CHR(13)
4470 L$=""
4475 PRINT @32,26:2
4480 C=1
4482 PAGE
4484 SET NOCASE
4490 END
5000 REM***********************************
5010 REM              ADD TO END OF TEXT
5020 REM***********************************
5030 PRINT "ADD -"
5040 INPUT A$
5050 IF A$="" THEN 5080
5060 A$=A$&M$
5070 T$=T$&A$
5075 GO TO 5040
```

```
5080 RETURN
6000 REM*********************************
6010 REM  CONVERT ALL LINES TO < 72 CHARACTERS
6020 REM*********************************
6030 C=1
6040 E=POS(T$,M$,C)
6050 L$=SEG(T$,C,E-C+1)
6060 IF LEN(L$)<=72 THEN 6150
6070 A=POS(L$," ",1)
6080 IF LEN(L$)-A<=72 THEN 6110
6090 A=POS(L$," ",A+1)
6100 GO TO 6080
6110 T$=REP(M$,C+A,0)
6120 E=C+A
6130 GOSUB 1530
6140 GOSUB 1160
6150 GOSUB 10340
6160 IF E<LEN(T$) THEN 6050
6170 RETURN
10000 REM************************************
10010 REM                SUBROUTINES
10020 REM************************************
10030 REM =========
10040 REM SUBROUTINE TO FIND FILE
10050 PRINT "ENTER FILE NUMBER TO ";A$;" -";
10060 INPUT F
10070 FIND F
10080 RETURN
10090 REM============
10250 REM SUBROUTINE TO MOVE UP 1 LINE
10260 IF C=1 THEN 10320
10262 E=C-1
10270 C=C-2
10280 A$=SEG(T$,C-1,1)
10290 IF A$=M$ OR C=1 THEN 10320
10300 C=C-1
10310 GO TO 10280
10320 RETURN
10330 REM=========
10340 REM SUBROUTINE TO MOVE DOWN 1 LINE
10350 IF E=>LEN(T$) THEN 10370
10352 C=E+1
10360 E=POS(T$,M$,C)
10370 RETURN
```

APPENDIX C

TEKTRONIX TO IBM PROGRAM

```
1 GO TO 900
4 D1=1
5 GO TO 600
8 GO TO 300
12 GO TO 400
16 GO TO 500
24 A$="PULL"
25 I=1
26 GO TO 100
28 A$="FUEL"
29 I=2
30 GO TO 100
32 A$="RPM"
33 I=3
34 GO TO 100
36 A$="SPEED"
37 I=4
38 GO TO 100
40 A$="SLIP"
41 I=5
42 GO TO 100
44 D1=0
45 D2=41
46 GO TO 600
48 D1=0
49 D2=32
50 GO TO 600
60 CALL "TERMIN"
96 REM*************************************
97 REM                  SUBROUTINES
98 REM*************************************
99 REM---------SUBROUTINE TO SET RANGE ON VARIABLES.
100 IF S1(I)=0 THEN 140
110 S1(I)=0
120 PRINT A$;" HAS BEEN DESELECTED."
130 RETURN
140 S1(I)=1
150 PRINT "ENTER MINIMUM VALUE OF ";A$;" TO RECALL: ";
160 INPUT S(I,1)
170 PRINT "ENTER MAXIMUM VALUE OF ";A$;" TO RECALL: ";
180 INPUT S(I,2)
190 RETURN
200 REM----DISPLAY VARIABLE SELECTION STATUS
210 IF S1(I)=1 THEN 240
220 PRINT A$;" HAS BEEN DESELECTED."
230 RETURN
240 PRINT A$;" MINIMUM = ";S(I,1);"    ";
          A$;" MAXIMUM = ";S(I,2)
250 RETURN
297 REM*************************************
298 REM             DISPLAY SEARCH PARAMETERS
299 REM*************************************
300 PAGE
310 RESTORE
```

```
320 FOR I=1 TO 5
330 READ A$
340 GOSUB 200
350 NEXT I
360 PRINT
370 PRINT "SEARCH PARAMETERS:";S$
380 RETURN
397 REM*********************************************
398 REM                ADD SEARCH PARAMETERS
399 REM*************************
400 PRINT "ENTER SEARCH PARAMETER TO ADD:";
410 INPUT A$
412 IF A$="" THEN 440
420 A$=A$&" "
430 S$=S$&A$
440 RETURN
497 REM*************************************
498 REM                  DELETE SEARCH PARAMETER
499 REM*************************************
500 PRINT "ENTER SEARCH PARAMETER TO DELETE:";
510 INPUT A$
520 X=POS(S$,A$,1)
530 IF X=0 THEN 550
540 S$=REP("",X,LEN(A$)+1)
550 RETURN
597 REM*************************************
598 REM (SEARCH PRINT,TRANSFER,DISPLAY)
599 REM*************************************
600 PRINT "ENTER FILE NUMBER TO SEARCH:";
610 INPUT F
620 PRINT "INSERT TAPE, HIT RETURN."
630 INPUT A$
640 FIND F
650 ON EOF (0) THEN 880
660 INPUT @33:A$
662 L$=""
670 I=1
680 B$=SEG(A$,(I-1)*6+1,6)
690 X=VAL(B$)
692 IF S1(I)=0 THEN 712
700 IF X<S(I,1) OR X>S(I,2) THEN 660
710 L$=L$&B$
712 I=I+1
720 IF I<=5 THEN 680
730 A$=SEG(A$,31,LEN(A$)-30)
740 B=1
750 X=POS(A$," ",B)
760 C$=SEG(A$,B,X-B)
770 X1=POS(S$,C$,1)
772 B=X+1
780 IF X1=0 THEN 810
800 L$=L$&C$
802 L$=L$&" "
810 IF B<LEN(A$) THEN 750
```

```
820 IF D1=1 THEN 850
830 PRINT @D2:L$
840 GO TO 660
850 PRINT @40:L$
860 INPUT %40:A$
870 GO TO 660
880 END
897 REM*****************************************
898 REM                              INITIALIZE
899 REM*****************************************
900 DELETE S,S1
910 DIM S1(5),S(5,2)
920 D1=0
930 L$=""
940 S$=""
950 PRINT @32,26:2
960 SET KEY
970 FOR I=1 TO 5
980 S1(I)=1
990 S(I,1)=-99999
1000 S(I,2)=99999
1010 NEXT I
1020 PRINT @37,0:19,255,255
1030 PRINT @41,11:1
1032 DATA "PULL","FUEL","RPM","SPEED","SLIP"
1040 END
```

APPENDIX D

AIM 65 PROGRAM

```
0000                            *=$B000
                        ;MONITOR COMMAND '5' BEGINS AT B000
B000   A9 54                    LDA #$54
B002   8D 12 A4                 STA $A412
                        ;DEFINES INPUT DEVICE AS TAPE RECORDER
B005   A9 00                    LDA #$00
B007   8D 34 A4                 STA $A434
                        ;DEFINES TAPE RECORDER AS RECORDER 1
B00A   A9 41                    LDA #$41
B00C   8D 2E A4                 STA $A42E
B00F   8D 2F A4                 STA $A42F
B012   8D 30 A4                 STA $A430
B015   8D 31 A4                 STA $A431
B018   8D 32 A4                 STA $A432
                        ;DEFINES FILENAME AS 'AAAAA'
B01B   20 EA ED                 JSR $EDEA
                        ;SETUP FOR INPUT AND GET SYN CHARS
B01E   20 2F E3                 JSR $E32F
                        ;SETS RECORDER ON, SEARCS
B021   A2 00                    LDX #00
B023   20 3B ED         MDATA   JSR $ED3B
                        ;LOADS A HEX CHAR FROM TAPE BUFFER TO ACC
B026   95 00                    STA 00,X
                        ;PUTS HEX CHAR IN ZERO PAGE
B028   E8                       INX
B029   E0 8D                    CPX #$8D
                        ;
B02B   D0 F6                    BNE MDATA
                        ;THERE ARE 8D BYTES PER EXP. UNIT
B02D   A2 00                    LDX #$00
B02F   B5 00            TRANS1  LDA $00,X
                        ;
B031   20 46 EA                 JSR $EA46
                        ;CONVERT TO ASCII, THEN OUTPUT TO TEKTRONIX
B034   E8                       INX
B035   E0 64                    CPX #$64
                        ;64 HEX IS NUMBER OF LAST VALID DATA BYTE
B037   D0 F6                    BNE TRANS1
B039   A2 70                    LDX #$70
                        ;70 HEX IS LOCATION OF FIRST CALIBRATION CON
B03B   B5 00            TRANS2  LDA $00,X
B03D   20 46 EA                 JSR $EA46
                        ;CONVERT TO ASCII, THEN OUTPUT TO TEKTRONIX
B040   E8                       INX
B041   E0 80                    CPX #$80
                        ;80 HEX IS LAST VALID CALIBRATION CONSTANT
B043   D0 F6                    BNE TRANS2
```

```
*=$B000 ;          MONITOR COMMAND '5' BEGINS AT B000
LDA #$54
STA $A412 ;        DEFINES INPUT DEVICE AS TAPE RECORDER
LDA #$00
STA $A434 ;        DEFINES TAPE RECORDER AS RECORDER 1
LDA #$41
STA $A42E
STA $A42F
STA $A430
STA $A431
STA $A432 ;        DEFINES FILENAME AS 'AAAAA'
JSR $EDEA ;        SETUP FOR INPUT AND GET SYN CHARS
JSR $E32F ;        SETS RECORDER ON, SEARCS
LDX #00
MDATA JSR $ED3B ; LOADS A HEX CHAR FROM TAPE BUFFER TO ACC
STA 00,X ;         PUTS HEX CHAR IN ZERO PAGE
INX
CPX #$8D ;
BNE MDATA ;       THERE ARE 8D BYTES PER EXP. UNIT
LDX #$00
TRANS1 LDA $00,X ;
JSR $EA46 ;       CONVERT TO ASCII, THEN OUTPUT TO TEKTRONIX
INX
CPX #$64 ;         64 HEX IS NUMBER OF LAST VALID DATA BYTE
BNE TRANS1
LDX #$70 ;        70 HEX IS LOCATION OF FIRST CALIBRATION CON
TRANS2 LDA $00,X
JSR $EA46 ;       CONVERT TO ASCII, THEN OUTPUT TO TEKTRONIX
INX
CPX #$80 ;        80 HEX IS LAST VALID CALIBRATION CONSTANT
BNE TRANS2;

JMP $E102 ;       ALL DONE. JUMP TO MONITOR
```

VITA

Douglas Robert DeVoe

Candidate for the Degree of

Master of Science

Thesis: A DATA MANGAGEMENT SYSTEM FOR A MICROCOMPUTER BASED TRACTOR DATA ACQUISITION SYSTEM

Major Field: Agricultural Engineering

Biographical:

Personal Data: Born in Ann Arbor, Michigan, December 26, 1956, the son of Dr. and Mrs. James R. DeVoe.

Education: Received Bachelor of Science in Agricultural Engineering at Oklahoma State University in 1980. Enrolled in Master's program in August, 1980 at Oklahoma State University; completed requirements for the Master of Science degree at Oklahoma State University in December, 1982.

Professional Experience: Student Member of The American Society of Agricultural Engineers and The Oklahoma Society of Professional Engineers. Graduate research assistant, Oklahoma State University, Agricultural Engineering Department, 1980-1982. Oklahoma Society of Professional Engineers Outstanding Student Achievement Award, 1980. American Society of Agricultural Engineers Southwest Region Student Paper Contest, first place, 1980.